no Fiche available

ESD-TR-77-23

This report recalled from DTIC
Per Lt Col Grewe, Lt Col Nealy ESD/TOIT ELC 7Apr80

SFEP SUBSYSTEM SPECIFICATION

Honeywell Information Systems, Incorporated
Federal Systems Operations
7900 Westpark Drive
McLean, VA 22101

October 1976

Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

This technical report has been reviewed and is approved for publication.

WILLIAM R. PRICE, Capt, USAF
Techniques Engineering Division

ROGER R. SCHELL, Lt Col, USAF
ADP System Security Program Manager

FOR THE COMMANDER

STANLEY P. DERESKA, Colonel, USAF
Deputy Director, Computer Systems Engineering
Deputy for Command & Management Systems

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-77-23	2. GOVT ACCESSION NO.
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): SFEP SUBSYSTEM SPECIFICATION	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): C. H. Bonneou J. J. Carnall M. F. Hall	8. CONTRACT OR GRANT NUMBER(s): F19628-74-C-0193
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Honeywell Information Systems, Incorporoted Federol Systems Operotions 7900 Westpark Drive, McLean, VA 22101	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Command and Management Systems Electronic Systems Division Hanscom AFB, MA 01731	12. REPORT DATE: October 1976
	13. NUMBER OF PAGES: 186
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Rleose; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Security
Secure Front-End Processor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This specification defines the technical requirements for hardware and software for a Secure Front-End Processor (SFEP). The SFEP is intended to serve as the front-end processor for Multics and other Honeywell 6000/Series 60 machines allowing the concurrent handling of multi-level classified traffic in a secure manner. The SFEP hardware elements include a Honeywell Level 6 computer, a Security Protection Module (SPM), a communication

Continued on reverse side

20. ABSTRACT (Cont.)

network interface unit, and a central system interface unit. The software elements include a security kernel, operating system, application and support software.

Preface

Air Force Systems Command terminated the effort which this document describes before the effort reached its logical conclusion. This specification has not been formally approved but was published in the interest of capturing and disseminating the computer security technology that was available at the time of the termination.

This specification describes the hardware and software necessary for the front-end processor subsystem of a secure general-purpose computer system (Multics). This subsystem includes a secure communications processor and a software security kernel for the processor which would be suitable for many minicomputer applications with security requirements.

Although the design specified for the front-end processor subsystem appears sound for the most part, the Air Force did not approve the specification. The Air Force review generated significant technical comments. The review noted inconsistencies within the specification and between this specification and the specifications of components of the subsystem. The review also noted that a lack of clarity and the use of undefined terms make the specification difficult to understand for the reader who is not familiar with the project. More specific technical comments can be found in an appendix to this specification.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF FIGURES

1.0 SCOPE

This specification establishes the performance, design, development, and test requirements for the Secure Front-End Processor (SFEP). The SFEP serves as the front-end processor for Multics and other Honeywell 6000/Series 60 machines allowing the concurrent handling of multi-level classified traffic in a secure manner. The SFEP shall consist of elements of the Honeywell Series 60 Level 6 commercial product, a Security Protection Module (SPM), a communication network interface unit, and a central system interface unit.

2.0 APPLICABLE DOCUMENTS

2.1 General Applicability

The following documents, including the revisions listed, form a part of this specification to the extent specified herein. In the event of conflicts between documents, the following precedence shall apply:

A. The Statement of Work shall take precedence over the Detail Specification.

B. The Detail Specification shall take precedence over all documents invoked herein.

C. Any documents invoked herein shall take precedence over all subsidiary documents.

2.2 Military Specifications and Standards

MIL-STD-130		Identification Markings of U.S. Military Property
MIL-STD-461A	1 Aug. 68	Electromagnetic Interference
MIL-STD-1000		Drawings, Engineering and Associated Lists
MIL-STD-1472A	15 May 70	Human Engr. Design Criteria for Military Systems, Equipment and Facilities
MIL-STD-810C		Environmental Test Methods, Military Standard
MIL-E-5400P		Electronic Equipment, Airborne, General Specification For
MIL-STD-454		Standard General Requirements for Electronic Equipment

2.2 Military Specifications and Standards (Continued)

MIL-STD-461A (3)	Electromagnetic Interference Characteristics, Requirements for Equipment
MIL-C-38999	Connector, Electrical
MIL-STD-1472	Human Engineering Design Criteria for Military Systems, Equipment and Facilities

2.3 Government Documents

DCA Circular 370-D195-2, 29 Jan 1975	Test and Alterations, DCA Autodin TEMPEST Category II Testing

2.4 Honeywell Documents

BG8249A1	Detail Specification, Part I, Ruggedized Level 6 Minicomputer
ESD-TR-76-366	Detail Specification, Part I, Security Protection Module (SPM)
ESD-TR-76-355	Detail Specification, Part I, 6000/ Series 60 Interface Unit

2.5 <u>Other</u>

F19628-74-C-0205	Secure Communications Processor Architecture Study	
Schiller, W. L.	Mar. 1975	The Design and Specification of a Security Kernel for the PDP-11/45, MTR-2934, MITRE Corporation, Bedford, MA.
Schroeder, M. D. and Saltzer, J. H.	Mar. 1972	"A Hardware Architecture for Implementing Protection Rings", <u>Comm. ACM 15</u>(3), 157-170.
Bell, D. E. and LaPadula, L. J.	July 1975	Secure Computer System: Unified Exposition and Multics Interpretation, MTR-2997.
F19628-74-C-0173	Prototype Secure Multics Specification	
F19628-74-C-0193	Security Kernel Specification for Secure Communications Processor.	

3.0 REQUIREMENTS

This specification covers both existing equipment and new equipment to be developed. The new equipment shall be designed in accordance with the requirements of this specification. The existing equipment shall meet the requirements of this specification.

3.1 SFEP Definition

This specification defines the hardware and software for a Secure Front-End Processor (SFEP) required to concurrently handle multi-level classified traffic between communication lines and a host computer (i.e., Multics and other Honeywell 6000/Series 60 machines) while enforcing the access rules defined by military security regulations.

The role of the SFEP in a secure, general purpose computer system is shown in Figure 1. The SFEP serves as the interface between the host computer, the user terminals, and an external communications network such as ARPANET. The SFEP hardware and software elements are identified in Figure 2.

The SFEP hardware shall consist of elements of the existing Honeywell Series 60 Level 6 minicomputer system, a to be developed 6000/Series 60 Interface Unit (IU), a to be developed network interface unit, and a to be developed Security Protection Module (SPM).

3.1 SFEP Definition (Continued)

The SFEP software shall consist of a kernel, operating system, application software, and support software. In general, all of this software is to be developed, however, whenever practical, the operating system, application and support software elements shall be based on existing, commercially available software packages.

3.1.1 SFEP Functional Definition

The primary function of the SFEP is to provide message handling between remote user terminals, a central computer system, and an external communications network. This message handling must be done in a secure manner, i.e., military security regulations must not be violated. The following paragraphs present the functional requirements for providing a secure environment and define the essential hardware and software functions.

3.1.1.1 Security Requirements

SFEP shall provide a secure environment (i.e., information access rules defined by military security regulations) for message processing. To provide this environment, SFEP shall support the concept of a reference monitor or security kernel. The security kernel shall enforce the authorized access relationships between subjects and objects through a combination of hardware and software. Subjects are active system entities such as a user or a process that can access system resources; and objects are passive system entities such as data, programs, and peripheral devices that can be accessed by subjects. It should be noted that devices can be subjects as well as objects.

The SFEP security kernel shall meet three essential design requirements: (1) the security kernel shall be invoked on every access to every resource -- complete mediation property; (2) the security kernel shall be protected from unauthorized alteration via concentric rings of protection -- isolation property; and, (3) the security kernel's correctness must be provable in a rigorous manner using a mathematical model as the basis for the criteria to be met -- verifiability property. The mediation and isolation requirements are defined in this section, the verification requirements are defined in 4.2.4.

3.1.1.1.1 Mediation Requirements

The complete mediation property of the security kernel requires that every use of any resource of SFEP (e.g., a memory location, a peripheral device, a communications line, a processor) is made only via kernel-enforced checks. The kernel maintains process access levels that are based on user clearance levels.

The access mediation within SFEP shall be implemented in terms of rules that control the access of subjects to objects. Every object shall be explicitly assigned a security attribute that shall represent the classification and categories of the object. Every subject shall be explicitly assigned a security access level that shall represent the clearance level and the set of access categories authorized for the subject. Every subject shall be explicitly designated as trusted or untrusted. A subject may be designated trusted if it has been verified analytically that its operations will not transfer information of a given classification into objects of lower classification, resulting in a compromise. Otherwise, the subject is designated untrusted. The rules that control the access of subjects to objects are as follows:

a. Before granting information-access, a subject's clearance level and access category set shall be compared with an information object's classification level and categories. The subject shall be allowed to read the element of information only if the clearance level of the subject is greater than or equal to the classification level of the object, and the subject is authorized access to the set of categories that are assigned to the object.

3.1.1.1.1 Mediation Requirements (Continued)

b. No subject not specifically designated as trusted shall have read-access to an object of a higher classification than one to which the subject concurrently has write-access; i.e., information from an object with a given classification may only be transferred into an object with an equal or higher classification.

3.1.1.1.2 Isolation Requirements

The SFEP security kernel shall be tamper proof, i.e., it must be protected from unauthorized alternation. This protection shall be provided via hardware-supported multiple execution domains (or states or modes) based on access privilege. This domain organization is termed a ring structure. When operating within a ring, a process shall only have access to the resources of that ring and of all rings of less privilege; the resources of all rings of more privilege (i.e., inner rings) shall not be directed accessible. Methods for crossing rings shall be such that the isolation property is maintained.

Within SFEP, a minimum of three rings of privilege shall be supported. These rings are named ring zero to ring n, with ring zero being the most privileged. Ring 0 shall be the domain of the security kernel. Untrusted subjects shall never occupy ring 0 directly; only controlled access to the kernel shall be allowed. The operating system shall occupy one of the outer rings. Although the operating system software itself is unverified, it is necessary to isolate it from application software in order to minimize the possibility of denial of service. Application software shall occupy rings outside the operating system.

3.1.1.2 System Overview

The message processing requirements shall be met with a computer system based on the Honeywell Level 6 minicomputer appropriately modified to meet the security requirements. Both hardware and software elements shall be provided.

The SFEP hardware shall consist of elements of the existing Honeywell Series 60 Level 6 minicomputer system, a to be developed 6000/Series 60 Interface Unit (IU), a to be developed network interface unit, and a to be developed Security Protection Module (SPM). The IU shall provide the interface between the Level 6 system and a central system using the direct channel interface. The network interface unit shall provide the interface between the SFEP and an external communications network such as ARPANET. The SPM shall support security/performance through hardware mediation of the interfaces of the non-secure components of the SFEP.

3.1.1.2 System Overview (Continued)

The relationship of the SFEP hardware components are shown on the SFEP subsystem block diagram in Figure 3. The configuration shown is primarily illustrative and does not represent any particular SFEP installation. In addition to the standard central processor, security protection module, memory, central system interface, and communications interface required for all SFEP applications, Figure 3 also includes a network interface and illustrates the potential for multi-processor configurations, supporting mass store, and operator's console and associated diskette. Although particular applications may not require the full functionality of the SFEP configuration shown in Figure 3, the system design must not preclude its inclusion.

The SFEP software shall consist of a kernel, an operating system, application software, and support software. The kernel software shall support the SPM in enforcing the access controls required for handling multi-level security. The operating system shall provide the resource management functions that do not impact system security. The application software shall provide terminal handling and network interface functions. Support software shall provide start-up, operator support, and troubleshooting functions.

3.1.1.2 System Overview (Continued)

Figure 4 presents the SFEP software functional block diagram illustrating a process structure which satisfies the primary SFEP functional requirement (i.e., secure message handling between terminals, the central system, and an external communications network). Isolated user processes are created by a trusted Answering Service process upon user log-on. Each user is given an isolated and protected virtual environment within which to communicate with the central system or the communications network. The SFEP security kernel is responsible for mapping and mediating all user process accesses to its virtual resources. The virtual resources of each process consists, as a minimum, of a single device (terminal) channel, a terminal handler, and buffer space. Depending on performance issues, user processes may also contain the central system interface handler (within the distributed kernel) to complete the connection to the corresponding central system user process or communicate indirectly via the kernel Interprocess Communication (IPC) mechanism to a separate trusted central system interface handler. The IPC function may also be used to allow the implementation of resource allocation policy mechanisms outside the kernel as untrusted processes running at system high (e.g., scheduler and memory manager). At user log-out, the Answering Service

3.1.1.2 System Overview (Continued)

process deletes the user process from the SFEP system and notifies the central system. The isolated virtual environment provided each user (enforced by the security kernel) within SFEP shall allow concurrent handling of multi-level security data in a manner which satisfies DoD security regulations.

The security requirements of mediation and isolation shall be met by providing the following functional capability.

3.1.1.2.1 Mediation Implementation

The SPM shall mediate all interactions between elements of the protected front-end processor. An SPM shall be associated with each processor of SFEP. Through its SPM, each processor shall communicate with the other processors, I/O devices, and memory. An I/O device shall communicate to memory through either the SPM of the processor that initiated its current operation or any other designated SPM within the system.

Mediation shall be implemented via a descriptor based virtual resource management scheme. To accomplish this, a mapping function, an access control function, a segmented memory structure, and a descriptor structure shall be supported.

3.1.1.2. 1.1 Mapping Mechanism

Fundamental to the concept of security is the management of virtual resources. The set of virtual names by which a process (defined to consist of a collection of resources and a state) invokes its resources defines a virtual environment for the process. Real resources shall be available to a process only through descriptors generated by the security kernel. The descriptors shall be used by the security kernel hardware to map the virtual resources of a process to the real resources of the system. The resource mapping function of SFEP shall be centralized within the Security Protection Module (SPM).

When a processor makes a memory reference, the memory address presented on the bus shall be intercepted by the SPM associated with that processor and shall always be treated as a virtual address. The SPM translates this virtual address into a physical memory address. The physical address is then presented to memory, and the appropriate read or write access is made. The data going to and from memory shall not be examined by the SPM.

In order to satisfy the universal mediation requirement in the simplest manner, the SPM would mediate each request by an I/O device to reference memory. However, due to performance difficulties imposed by the bus architecture of SFEP, the SPM's active mediation

may cause unacceptable performance loss in high I/O bandwidth applications. Thus, SFEP shall support two forms of I/O mediation:

Devices designated as premapped shall reference memory without mediation by the SPM; memory addresses are absolute. Premapped I/O mediation shall be performed completely at I/O initiation time by the SPM; that is, the SPM shall validate at I/O initiation all resulting I/O device memory accesses. At premapped I/O initiation, the virtual address associated with the transfer is delivered to the SPM. After suitable checking, the address shall be mapped by the SPM to an absolute memory address and delivered to the device. Transfer of data will occur directly between the device and memory. Figure 5 functionally presents premapped I/O flow.

The SPM shall mediate each request by an I/O device designated as mapped to reference memory. Mapped I/O devices shall reference memory using only virtual addressing. At mapped I/O initiation the virtual address associated with the transfer is delivered to the SPM for set up, and then is delivered to the device as a virtual address. The address of each item of data transferred shall be delivered to the SPM for mediation (i.e., mapping and checking). Each address delivered to the SPM shall be accompanied by the identification of the transferring device in order to allow SPM mediation to occur. Figure 6 functionally presents mapped I/O flow.

3.1.1.2.1.2 Access Control

The access privilege of a process shall be controlled by the SPM through comparison of an effective ring number generated by the SPM during address formation and pertinent access control fields contained in the descriptor controlling access to the accessed resources. The access control rules are specified in detail in 3.7.1.3.

3.1.1.2.1.3 Segmented Memory

Access control shall be based on compartmentalizing all of the stored information contained within SFEP memory into discrete packages called segments each having associated with it a set of access attributes that describe the fashion in which each process is permitted to reference the contained information. Addressing within SFEP shall be "two-dimensional"; that is, stored information shall be addressed via an ordered pair (name, offset), where "name" is the segment designator and "offset" is the word number in the corresponding linear memory for that segment.

SFEP shall provide a maximum of 512 segments consisting of 2K 16-bit words. These segments may be subdivided into 16 subelements called pages consisting of 128 16-bit words. Access control shall be to the segment level only.

3.1.1.2. Descriptor Mechanism
1.4

Every resource that is allocated to a process shall be represented by descriptors. Descriptors are constructed by the security kernel and are structured in memory for use by the SPM. The descriptor structure is the prime data base for the state of allocation of the system resources. Descriptors shall be supported for both memory and I/O devices. Descriptor Base Roots (DBRs) used by the SPM to establish the set of descriptors for a process shall also be supported.

Memory descriptors provide the mechanism for the SPM to convert virtual memory addresses presented by requestors (i.e., CPU's or mapped I/O devices) to absolute memory addresses. In addition, the memory descriptors provide the access control mechanism for the SPM to verify that the process has the required access privilege to the referenced memory location. The SPM shall insure that descriptor copies within the SPM reflect the memory originals and that the memory originals are updated to reflect the changes made to the SPM's copies (e.g., used and modified bits).

I/O descriptors provide the mechanism for the SPM to convert virtual device addresses presented by requestors (i.e., CPU's) to absolute device addresses. In addition, the I/O descriptors provide the access control mechanism for the SPM to verify that the process has the required access privilege to the referenced device. The SPM shall insure that descriptor copies reflect the memory originals and that the

3.1.1.2. <u>Descriptor Mechanism</u> (Continued)
1.4

memory originals are updated to reflect the changes made to the SPM's copies.

Details of what information shall be contained in the descriptors is specified in 3.7.1.3.

3.1.1.2. <u>Isolation Implementation</u>
2

Isolation shall be based on a ring structured architecture. To assure isolation, several special functions shall be provided. These are functions associated with ring crossings and privileged functions for controlling the SPM. The ring crossing functions include: CALL/RETURN, TRAP/TRAP RETURN, and INTERRUPT/INTERRUPT RETURN. The privileged functions include: DISPATCH, SELECTIVE DESCRIPTOR INVALIDATION, and SPM T&D.

The privileged functions directly accessing the SPM shall, of course, be limited to the kernel software. To reduce mediation overhead, the SPM shall be addressed as an absolute Programmed Input/Output (PIO) type device. PIO refers to device control functions as opposed to normal data transfer functions. The absolute channel number used to represent the SPM shall be assignable at system configuration time. This interface would restrict the kernel from performing I/O with other devices whose virtual addresses are identical to the SPM absolute

3.1.1.2.2 Isolation Implementation (Continued)

address and which utilize the same function codes as selected for the SPM interface. To eliminate this restriction, the SPM function codes shall be restricted to those not utilized by other devices.

Functions to support the isolation property may be distributed between the processor, the SPM, and kernel software.

3.1.1.2.2.1 CALL and RETURN

CALL and RETURN instructions shall be provided by the processor and be recognized by the SPM. A CALL is normally used to transfer to inner ring procedures to accomplish more privileged operations than those allowed at the current ring, and a RETURN is used to return from an inner ring procedure back to the outer ring from which the call originated. Since the CALL instruction may change the current ring (R_{cur}) to a lower number and thus place the processor in a more privileged state, the SPM must guarantee that entry into the inner ring is tightly and completely controlled by that inner ring. Thus, the SPM must check that an inner ring may only be entered at specific locations within specific procedures. Outward calls to untrusted code shall not be allowed since a potential for security compromise would exist.

3.1.1.2. CALL and RETURN (Continued)
2.1

The SPM shall mediate the virtual entry point address placed on the address bus by a process executing the CALL instruction in the same manner as for a normal memory reference. However, unlike normal memory references, no mapping is performed. That is, the SPM shall only validate the virtual entry point address (i.e., caller has execute access) and compute a new value of R_{cur} as specified in 3.7.1.3.

If the effective ring number of the caller is outside the call bracket, the SPM shall trap. Similarly, in order to control entry to the called procedure, the SPM shall insure that only location zero of the called procedure (defined by the direct page or segment descriptor) be a valid entry point. If a call to an invalid entry point is attempted, the SPM shall trap. It should be noted that the entry point limitation is the degenerate example of a call limiter restriction specifying the maximum offset within a segment or page to which a call can transfer to be zero. The SPM shall perform all required access checks prior to changing the current ring of execution and allowing the virtual entry point address to be inserted into the CPU program counter.

3.1.1.2 CALL and RETURN (Continued)
2.1

The security kernel shall provide process local stack space (stack per ring) to support argument passing and to satisfy reentrancy requirements.

Arguments shall only be passed via the stack mechanism. The called function shall be responsible for performing argument validation including caller stack validation. The caller's ring number shall be provided to the called procedure to support general argument validation. A special VALIDATE instruction shall be supported by the SPM. VALIDATE shall indicate the access rights to a memory area for a given level of privilege (i.e., specified ring number). Each called procedure shall contain a check on the caller's level of privilege; if the caller's ring is the same as the called, the argument validation checks are to be bypassed.

3.1.1.2. Trap and Trap Return
2.2

Traps are software initiated events (either intentional or unintentional) to which the processor responds by saving the current state of the processor in such a way that it can later be restored, and transferring control to a specified memory location. In a secure system, many traps occurring in a given ring are best handled by software executing in that ring. Some traps, however, such as those generated by the SPM, are best handled by inner ring software. SFEP shall allow trap handling to be performed in the ring appropriate to the type of trap; however, initial processing must be performed within the security kernel.

Upon the occurrence of a trap, CPU firmware shall select an area for storage of information about the state of the process and an entry point to a service procedure. Since the CPU provides a single storage area/entry point per trap type/trap occurrence and traps may occur within any ring, all trap storage areas and service procedures shall reside within the security kernel (i.e., ring 0). In addition, the security kernel shall be solely responsible for specifying all trap vectors and storage area pointers for each process.

3.1.1.2. Trap and Trap Return (Continued)
2.2

The trap save area shall contain sufficient information to allow the service procedure to determine the corrective action required to resume normal process execution. Included within the trap save area shall be the ring number of the process (R_{cur}) at the time of trap.

The trap save areas shall be process local, not global. The security kernel shall be responsible for insuring that each process has sufficient real memory to support its virtual trap save area free list upon process dispatch. A minimum of three save areas shall be provided per running process in order to prevent a trap save area run-out interrupt from occurring.

Upon the occurrence of a trap, the SPM shall force $R_{cur} = 0$, and shall translate the hardware-generated virtual addresses which specify the trap handler entry point and the trap save area.

The trap return instruction (RTT) shall restore the state of the process from the trap save area as modified by the associated service procedure. The restored state shall include the previously saved value of R_{cur}. The SPM need not check this ring number since

3.1.1.2. Trap and Trap Return (Continued)
2.2

it is issued by the kernel only. The trap return instruction shall be a privileged instruction; i.e., RTT shall be executable only within ring 0 (attempted execution of RTT within an outer ring shall result in a CPU-generated trap).

For those traps for which it is desirable to service in an outer ring, the kernel service routine (in this case, a trap executive), shall copy the non-security sensitive trap data from kernel space to an area accessible to the actual handler in an outer ring. Prior to the copy, the security kernel shall insure that the level of privilege (ring) of an outer ring trap handler is greater than or equal to the level of privilege in effect at the time the trap occurred. The kernel service routine shall then transfer control to the outer ring trap handler via the RETURN order. Upon completion of trap servicing, the outer ring trap handler shall CALL the kernel to perform any necessary modifications to the original trap save area data (in kernel space) and instruct the kernel to issue the RTT instruction on its behalf.

3.1.1.2.2.3 Interrupt and Interrupt Return

Unlike traps, which are synchronous with and in some sense caused by the currently executing process, interrupts are generally unrelated to the current process (or at least asynchronous with it). However, like traps, the processor responds to an interrupt by saving the current state of the processor in such a way that it can later be restored, and transferring control to a specified memory location Thus,SFEP requirements for interrupts are similar to those for traps.

3.1.1.2. Interrupt and Interrupt Return (Continued)
2.3

Upon the occurrence of an interrupt of higher level than the currently executing process level, CPU firmware shall save the current state of the processor using the current level's interrupt save area pointer, and restore the new state of the processor using the interrupting level's save area pointer. The identity of the interrupting channel shall be saved within the interrupting level's save area. Since the CPU provides a single storage area per interrupt level and interrupts may occur within any ring, all interrupt save areas and associated interrupt handlers shall reside within the security kernel (i.e., ring 0). In addition, the security kernel shall be solely responsible for specifying all interrupt level vectors for each process.

The interrupt save area shall contain sufficient information to allow resumption of execution of the interrupted process. Included within the save area shall be the ring number of the process (R_{cur}) at the time of interrupt.

The process ring number in the interrupting level's save area shall be ring 0 to force interrupt handling within the security kernel. It should be noted that the identity of the interrupting channel is absolute; it shall be the responsibility of the security kernel to perform the inverse mapping of an absolute to virtual device identification in order to notify the process associated with interrupting device (i.e., device wakeup function).

3.1.1.2. <u>Interrupt and Interrupt Return</u> (Continued)
2.3

Upon the occurrence of an interrupt, the SPM shall force $R_{cur} = 0$ in order to allow saving and restoring of processor state (under firmware control) within kernel space. At the completion of the firmware controlled interrupt processing, the interrupt handler shall be entered with kernel privilege as specified in the interrupting level's save area (new value of R_{cur} is to be transmitted to the SPM).

The interrupt return shall be accomplished by the level change instruction (LEV) which shall restore the state of the processor from the interrupted level's save area. The restored state shall include the previously saved value of R_{cur}. The SPM need not check this ring number since it is issued by the kernel only. The LEV instruction shall be a privileged instruction; i.e., LEV shall be executable only within ring 0 (attempted execution of LEV within an outer ring shall result in a CPU-generated trap).

3.1.1.2. Dispatch
2.4

The security kernel shall include a dispatch function to be used to initiate a new process, i.e., a ready process is to have its state changed to running. Functionally, a process shall notify its associated SPM that a new descriptor tree structure is to be utilized for access mediation and the previous descriptor structure is to be discarded. Details of the dispatch function are specified in 3.7.1.3.1.

3.1.1.2. SELECTIVE DESCRIPTOR INVALIDATION
2.5

It shall be a kernel responsibility to insure that changes made to the descriptor structure in memory for the currently executing process are reflected in the fast access copies retained by the SPM within its Fast Access Descriptor Store (FADS). This shall be accomplished via selective descriptor invalidation orders issued by the kernel to the appropriate SPM. Upon receipt of such orders, the SPM shall retrieve from memory the updated descriptor describing the resource as necessary for mediation of subsequent resource accesses; the invalidated descriptor within the FADS shall be overwritten by the newly retrieved descriptor. Details of the SELECTIVE DESCRIPTOR INVALIDATION are specified in 3.7.1.3.3.1.

3.1.1.2. SPM T&D
2.6

Within SFEP, the SPM test and diagnostic function shall be performed by a trusted process. SPM T&D shall be performed periodically to minimize the possibility that an SPM hardware failure results in the undetected loss of the SFEP information protection mechanism. The entire functionality of the SPM shall be thoroughly exercised via

3.1.1.2. SPM T&D (Continued)
2.6

various restricted I/O operations. Any detected SPM failure shall be immediately reported to the SFEP operator/security officer.

3.1.2 Functional Interface Definition

The interfaces between the SFEP and the outside world (i.e., external to the SFEP), interfaces between the SFEP hardware components, and interfaces between the SFEP software components are specified below.

3.1.2.1 Interfaces Between the SFEP and the Outside World

SFEP interfaces to the outside world shall consist of terminal communications lines, a central host computer interface, a communications network interface, and an operator interface for support functions.

3.1.2.1.1 Terminal Interface

SFEP shall provide the control of the communication lines and remote terminals. The SFEP shall be capable of supporting terminals operating in both half-duplex and full-duplex modes and at a variety of speeds (including 110, 134.5, 150, 300, 1200, 2400, 4800, and 9600 bps). The number of terminals shall be modularly expandable up to a maximum of 256 terminals. Terminals supported shall include TTY Model 40 and all those supported by the current Multics system.

3.1.2.1. Terminal Interface (Continued)
1

In particular, the following terminal interfaces shall be supported:

A. Asynchronous (transmits even parity)

a) Bell 103 interface, 7 bit ASCII, at 110, 150, and 300 baud.

b) Bell 103 interface, 6 bit EBCDIC, with odd parity at 134.5 baud.

c) Bell 202C6 interface, 7 bit ASCII, at 1200 baud (using common protocol established in MIT MAC-M-370).

d) Bell 202C5 and Bell 202C6 interfaces, at 1200 baud (using Terminet 1200 line protocols).

e) Vadic 3400 interface, 7 bit ASCII, at 1200 baud.

B. Synchronous (transmits odd parity)

a) Bell 208A and Bell 208B, 7 bit ASCII, at 4800 baud (using G115 protocol).

b) Bell 209A, 7 bit ASCII, at 9600 baud (using G115 protocol).

3.1.2.1.2 Central System Interface

SFEP shall interface with the central system via the Direct Channel Adapter (DCA) connected to the 6000 IOM. One central system interface shall support all SFEP configurations including multi-computer or multi-processor configurations. Throughput of the central system interface shall only be constrained by 6000 IOM capability.

3.1.2.1. <u>Central System Interface</u> (Continued)
2

The central system interface shall support the "scatter-gather" I/O concept. That is, discontiguous areas of both SFEP and central system memory may be accessed (data transferred) as a result of a single I/O command from the SFEP CPU. The interface shall support two data formats to convert 36-bit central system words to 16-bit SFEP words : ASCII and binary. While operating in the ASCII mode, the interface shall interpret a central system word as consisting of two SFEP words; i.e., four 8-bit characters per central system word. While operating in the binary mode, each central system word shall be interpreted as consisting of two and a fraction SFEP words. In the binary mode, only multiples of four central system words need be transferred allowing a four-to-nine mapping.

3.1.2.1.3 Network Interface

SFEP shall be capable of providing an interface to networks such as IDN II (Autodin II), SATIN IV, or the ARPANET. In particular, SFEP shall allow for the connection of a network interface implementing the American National Standards Institute (ANSI) Advanced Data Communication Control Procedures (ADCCP) - Independent Numbering, X3934/589, Draft 3, 12 December 1974. Specifically, the operational class defined as Primary-to-Primary, full duplex shall be used. The interface shall be capable of operating at the standard communications circuit speeds of 110 bps to 56 kbps, and shall include all responses and commands defined for the Primary-to-Primary Selective Reject Exception Recovery Class of Procedures. However, the capability to intermix the Set Asynchronous Response Mode (SARM) and the Set Asynchronous Response Mode Extended Format (SARME) commands on any one circuit shall not be implemented. Any given circuit shall optionally be capable of operating SARM or SARME but not both. The command for the mode selected shall only serve as a reset of the protocol.

3.1.2.1.4 Support Functions

SFEP shall provide utility-type functions to support successful front-end operations. These utilities shall include the following:

Bootload - SFEP shall be loaded from the central system via the central system interface.

Operator Interface - SFEP design shall allow the inclusion of an optional operator's console to support configuration management, debug, and T&D functions.

Debug - SFEP shall provide a debug capability including system dump via the operator's console.

3.1.2.1.4 Support Functions (Continued)

T&D - The ability to run Test & Diagnostic software offline shall be supported. T&D software shall be loaded via an optional diskette interface and shall interface with the operator's console.

Recovery/Restart - SFEP design shall not preclude the inclusion of automatic recovery/restart capability. In addition, SFEP design shall not preclude providing continued support to terminals in the event of a central system failure including routing to alternate hosts if allowed by the configuration.

3.1.2.2 Interfaces Between the SFEP Hardware Components

The primary interface between the SFEP hardware components shall be the Level 6 asynchronous bus.

A private interface between the SPM and the CPU may be used as required.

3.1.2.3 Interfaces Between the SFEP Software Components

Two types of interfaces are specified between the software elements. One is the interface between the untrusted processes and the distributed kernel. The other is the interface between processes.

The interface between the distributed kernel and the untrusted process software shall be a rigorously defined interface with access only through the Call/Return mechanism. The interface is precisely defined in the Security Kernel Specification for Secure Communications Processor.

The interface between processes shall be through the Interprocess Communication (IPC) mechanism. The IPC is specified in 3.7.2.2.1.

3.1.3 Major Component List

3.1.3.1 Hardware

The SFEP hardware shall consist of the following major components:

- Central Processing Unit (CPU)
- Security Protection Module (SPM)
- Memory
- Bus
- Communication Controllers (MLCP)
- 6000/Series 60 Interface Unit (IU)
- Inter System Link (ISL)
- Peripheral Device Controllers
- Network Interface Controller

3.1.3.2 Software

The SFEP software shall consist of the following major components:

- Security Kernel
- Operating System
- Application Software
- Support Software

3.1.4 Government Furnished Property List

Not applicable.

3.1.5 Government Loaned Property List

Not applicable.

3.2 SFEP Characteristics

3.2.1 Performance

Performance degradation of SFEP shall not exceed 25% of the performance delivered by an equivalent nonsecure front-end processor system (i.e., without the security protection mechanism) achieving 800,000 memory references per second.

3.2.1 Performance (Continued)

SFEP shall have a minimum throughput capability of 10K char/sec. This requirement results from assuming a maximum configuration of 256 terminals operating at an average rate of 300 baud (including sufficient margin).

3.2.2 Physical Characteristics

Although a commercial version can be provided, all references within this specification pertain to a ruggedized unit.

The housing shall consist of a cast chassis capable of holding 10 modules. A system may contain more than one chassis and provisions shall be made to attach additional, identical housings to the top and bottom of the basic unit. Provisions shall also be made for interconnecting the bus boards of additional chassis attached in the above manner.

The front surface of the housing shall contain the control panel and an air inlet. All external connectors as well as air exhaust will be on the back surface.

3.2.2.1 Weight

The total weight of a ten card SFEP computer shall be less than 150 pounds.

3.2.2.2 Outline Dimensions

Envelope dimensions shall include all protuberances (e.g., handles and connectors) except guide pins. See Figure 9.

3.2.3 Reliability

Design considerations and part selection shall be sufficient to assure that the equipment meets or exceeds its reliability requirements over its useful life.

3.2.3.1 Mean-Time-Between-Failures (MTBF)

Any single 10 board chassis configuration of the SFEP shall have calculated MTBF of 2800 hours minimum based on Honeywell commercial parts failure rates.

3.2.3.2 Probability of Failure Induced Security Compromise

As a design goal, the SFEP shall exhibit a probability of less than 0.000001 per hour that hardware failure will result in the undetected loss of secure data protection functions. The probabilistic measure of security compromise shall be established by analysis using failure rate data as specified in Paragraph 3.2.3.1.

3.2.3.3 Useful Life

The useful life of the equipment shall be 10 years when operated and maintained in accordance with the provisions of this specification.

3.2.4 Maintainability

The unit shall be designed with maintainability as a prime factor. The equipment shall comply with the following maintenance requirements.

A. The equipment shall be capable of being repaired with no special fixtures or tools.

B. Design configuration (layout) shall, to the maximum extent possible, be planned to facilitate the sequence of maintenance procedures (testing sequences, disassembly, adjustments, alignments, etc.) to eliminate backtracking redundant operations, or awkward activity by maintenance personnel.

3.2.4 Maintainability (Continued)

C. Line replaceable units shall not require adjustment or calibration in connection with replacement. (See the following paragraphs for LRU definition.)

D. The Mean-Time-To-Repair (MTTR) to plug-in card level, including fault isolation, removal, replacement and verification shall not exceed 60 minutes.

E. Plug-in card and power supply removal and replacement shall be no more time consuming than for the commercial equivalent.

F. The equipment shall be designed such that no preventative maintenance, other than normal cleaning is required.

G. External connectors shall be of the quick disconnect type, per MIL-C-38999 with a minimum of one inch (1") spacing provided between adjacent connectors.

3.2.4.1 Maintenance Concept

The equipment shall be designed to be maintained in accordance with established Air Force practices, policies and procedures. The existing three (3) levels of Air Force maintenance shall be used: organizational, field, and depot. The equipment shall be considered to meet the following definitions of subsystem and LRU.

The computer shall be defined as a subsystem for maintenance purposes.

Line replaceable units of the computer (subsystem) are:

- All 15" x 16" circuit cards (CPU, SPM, memory, MLCP, etc.)
- Power supply
- Front panel assembly

3.2.4.1.1 Organizational and Field Levels of Maintenance

For ground installations, organizational level and field level maintenance are synonymous and will consist of isolation of failures to the LRU. The LRU will then be sent to a depot for troubleshooting and repair. In cases where the failure cannot be isolated to an LRU or where the failure is associated with the housing assembly, the subsystem shall be replaced and the faulty subsystem shall be sent to the depot.

3.2.4.1.2 Depot Level Maintenance

Depot level maintenance consists of isolation and repair to the part level.

3.2.5 Environmental Conditions

The unit shall be designed to meet all functional specifications while subjected to the operational environments specified below in any combination and after being subjected to the non-operational environmental conditions specified below.

3.2.5.1 Temperature

Operating: Continuous $0^{\circ}C$ to $52^{\circ}C$

Non-Operating: $-62^{\circ}C$ to $+85^{\circ}C$

3.2.5.2 Altitude

In accordance with MIL-E-5400P, Paragraph 3.2.24.2 for Class 1A equipment with exception that the altitude requirement is 0 to 8,000 feet for continuous operation and 0 to 50,000 feet for exposure in a non-operating condition. (NOTE: It is anticipated that a MIL-STD-704A 400 Hz aircraft power supply with integral high volume fans will be developed at a later date. This supply would permit operation at higher altitudes.)

3.2.5.3 Humidity

The equipment shall maintain the specified performance when exposed to a relative humidity of 95% for both continuous and intermittent periods, including conditions wherein condensation takes place in and on the equipment in the form of both water and frost. Humidity tests shall be run per method 507, procedure IV of MIL-STD-810C (5 each, 24 hour cycles). A requirement established by this Detail Specification is that circuit boards shall be conformally coated. Uncoated circuit boards shall be available as a standard option to those users who specifically request uncoated boards. However, humidity qualification tests shall be run on a unit with both uncoated and conformally coated boards.

3.2.5.4 Vibration

The unit shall be capable of withstanding the following vibration spectrum without isolators:

5-2000 Hz, 2g peak per MIL-E-5400P curve IIA. (See Figure 10.)

With isolators, the unit shall be capable of withstanding the following vibration spectrum:

5-2000 Hz, 10g peak per MIL-E-5400P, curve IVA. (See Figure 10.)

3.2.5.5 Shock

3.2.5.5.1 Equipment

While isolated the equipment shall not suffer damage or functional failure when subjected to 18 impact shocks of 15g consisting of three shocks in opposite directions along each of three mutually perpendicular axes. Each shock impulse shall have a time duration of 11 $\pm$ 1 milliseconds. The maximum acceleration shall occur at approximately 5 1/2 milliseconds.

3.2.5.5.2 Mounting Base (Crash Safety)

With excursion stops or bumpers in place and with maximum rated load applied in a normal manner, the mounting base, individual isoaltors, or other attaching devices shall withstand at least 12 impact shocks of 30g, consisting of 2 shocks in opposite directions along each of 3 mutually perpendicular axes. Each shock impulse shall have time duration of 11 $\pm$ 1 milliseconds. The "g" value shall be within $\pm$ 10 percent when measured with a 0.2 to 250 Hz filter, and maximum "g" value shall occur at approximately 5 1/2 milliseconds. Bending and distortion shall be permitted; however, there shall be no failure to the attaching joints and the equipment or dummy load shall remain in place.

3.2.5.5.3 Bench Handling

In accordance with MIL-STD-810B, Method 516.1, Procedure V, Non-operating condition.

3.2.5.6 Explosive Conditions

Unit shall not produce exposed arcs in normal operation.

3.2.6 Transportability

Transportability shall be considered when formulating the design of the SFEP. MIL-P-9824 shall serve as a guide.

3.3 Design and Construction

3.3.1 Materials, Parts, and Processes

Materials, parts, and processes shall conform to the requirements of MIL-E-5400 when practical or unless otherwise restricted herein. Design and application considerations, as well as economic factors, shall govern the selection of and use of materials, parts, and processes.

3.3.1 Materials, Parts, and Processes (Continued)

HIS materials, parts, processes, and controlling specifications used for the existing Level 6 design and Aero FMS's and FPS's use for the SFEP design shall be considered approved for the SFEP upon verification of data substantiating that the unit will perform satisfactorily in the specified environment. In addition, the following paragraphs identify specific requirements and limitations in the use of materials, parts, and processes.

3.3.1.1 Hardware

3.3.1.1.1 Materials

3.3.1.1.1.1 Aluminum

Aluminum alloys 2020 and 7178 shall not be used for structural applications. Bare aluminum alloys 2024-T4 and 7075-T5 shall not be used in corrosion susceptible areas without suitable protective finishes. 7075-T6 shall not be used in fatigue critical applications as plate or as extrusions in sections greater than 0.25 inches.

3.3.1.1.1.2 Elastomeric Materials

Elastomeric components shall utilize only those elastomers which have adequate resistance to aging, ozone, heat aging, low temperature embrittlement and reversion, either temperature or moisture-temperature induced.

3.3.1.1.1.3 Wire

Wire used in all SFEP new designs shall conform to the following specifications:

A. 300V, Single conductor - FMS 40052

B. 300V, Shielded - FMS 40022

C. 600V, Single conductor - FMS 40053

D. 600V, Shielded - FSM 40051

3.3.1.1.1.4 Conformal Coatings

If required, printed circuit cards shall be conformally coated per FPS 18035, Type V.

3.3.1.1.2 Processes

3.3.1.1.2.1 Soldering

Electrical soldering practices shall be in accordance with FPS 18167. Certification of soldering operators is required.

3.3.1.1.3 Parts

3.3.1.1.3.1 Parts Selection and Standardization

Electronic part types for all new SFEP circuit designs shall be selected from the Level 6 Standard Parts List. All other electronic parts are non-standard. Selection, qualification and screening criteria applicable to non-standard parts shall be in accordance with Parts Control Program Requirements of the Honeywell SCOMP Reliability Program Plan. SCOMP is an acronym for Secure Communications Processor and SFEP is one application.

3.3.1.1.3.2 Electrical Connectors

Outside-world connectors of the SFEP shall meet the requirements of MIL-C-38999.

3.3.1.2 Programming Standards and Conventions

For each program, routine, subroutine or function developed, the structured programming concepts of top-down hierarchial design, top-down construction and restricted control logic structures shall be observed.

Programs shall be designed in a hierarchical manner and the levels of the hierarchy shall correspond to the levels of control of the program tasks. An essential characteristic is that each level of the program design shall be logically complete. Control shall proceed downward through the hierarchy; looping back through the structure shall not be permitted. Program modules shall be programmed at the highest level in the program hierarchical organization before modules of the succeeding levels.

3.3.1.2.1 Verification

To simplify verification of the security kernel software, it is anticipated that additional programming standards and conventions will be required. For example, if a compiler is to be used to convert the implementation language representation to machine language, an allowable subset of the high-order language need be established.

3.3.1.2.2 Consistency

To insure consistency among the various program modules, programming rules and conventions shall be established. For example, each module should have only one entry point; similarly, each module should have at most two exit points, one to handle error exits and one for normal exits.

3.3.1.2.3 Performance

To achieve SFEP performance requirements, programming rules and conventions shall be established. In particular, the issue of degree of process locality shall be addressed. Designers shall avoid specifying multiple rings unless necessary because each ring in which a subsystem executes adds segments to a process. Similarly, procedure segments and

data segments belonging to the same ring that are to have the same access controls shall be bound together. Procedure (and data) segment binding shall be done on the basis of the typical flow of control through them (or by references to them) rather than by their functional similarity.

3.3.2 Electromagnetic Compatibility

Electromagnetic compatibility criteria for the SFEP shall be in accordance with the emissions and susceptibility test requirements of MIL-STD-461A, Notice 3, 1 May 1970 for Class A3 equipment. Applicable requirements are:

CE03 - Conducted Emissions, power lines.

CE04 - Conducted Emissions, signal lines.

CS01 - Conducted Suscept., power lines, AF.

CS02 - Conducted Suscept., power lines, RF.

CS06 - Conducted Suscept., power lines, transient.

RE02 - Radiated Emissions, electric field.

RS03 - Radiated Suscept., magnetic induction field.

RS03 - Radiated Suscept., electric field.

3.3.2.1 <u>TEMPEST</u>

TEMPEST criteria for the SFEP shall be as specified by DCA Circular 370-D195-2.

Electric Field Space Radiation

Power Line Conduction

Black Signal Line Conduction

Red Signal Line Conduction

3.3.2.2 <u>Bonding</u>

Bonding shall comply with requirements of MIL-B-5087. An electrical bond is any fixed union existing between two objects that results in electrical conductivity between the objects. All permanent bonding inside the SFEP must be accomplished by direct metal to metal contact of properly prepared surfaces.

3.3.2.2.1 <u>Jumpers</u>

Bonding jumpers may be used to connect the SFEP chassis to rack structure if shock isolators are required. Such jumpers shall be at least 0.030 inches thick and have a length to width ratio no greater than 5 to 1.

3.3.2.2.2 <u>Bonding Surface Preparation</u>

Surface preparation for an electrical bonding shall

3.3.2.2.2 Bonding Surface Preparation (Continued)

be accomplished by removing all anodic film, grease, paint or other high resistance material from the immediate area to insure negligible RF impedance between adjacent metal parts. Chromate conversion finishes shall not be removed.

3.3.2.2.3 Removable Panels

Removable panels may be bonded through RFI gasket.

3.3.3 Identification and Marking

Identification and marking for Aero designed hardware shall be in accordance with MIL-STD-130. Identification and marking for BCO designed hardware shall be per BCO standards. If existing designs do not meet those requirements, it shall be documented and corrective action shall be taken if necessary.

3.3.4 Workmanship

Workmanship of Aero fabricated hardware shall be in accordance with the applicable portions of UED 23036. BCO workmanship standards shall apply to BCO fabricated hardware. General workmanship shall be of high quality to assure compliance with specification requirements including the service life requirement.

3.3.5 Interchangeability and Replaceability

3.3.5.1 General

Mechanical and electrical interchangeability shall exist between like assemblies, subassemblies, and replaceable parts regardless of manufacturer or supplier. Interchangeability, as used here, does not mean identity, but requires that a substitute of like assemblies, subassemblies, and replaceable parts may be easily effected without physical or electrical modifications to any part of the equipment or assemblies including cabling, wiring, and mounting.

3.3.5.2 Module Interchangeability

All major assemblies of the SFEP shall be replaceable and interchangeable without electrical adjustment or calibration.

3.3.6 Safety

A. The SFEP shall be designed to combine maximum safety and stability, avoiding sharp edges, protrusions, obstructions and any other mechanical or physical features which constitute a hazard in accordance with MIL-STD-1472, Sections 5.13.4 and 5.13.5.

3.3.6 Safety (Continued)

B. Means to prevent accidental exposure to electrical voltages in excess of 32 volts including potentials on charged capacitors shall be provided. Equipment shall be designed and constructed so that all external electrical parts will be at ground potential at all times. Where access is required for adjustment purposes during the normal operation of equipment, provide doors, covers or plates for compartments which employ an interlock to remove potentials in excess of 150 volts. Wiring shall be routed so as not to have "hot" leads on male connector pins upon plug disconnection. Provisions for avoiding electrical hazards will be designed in conformance with MIL-STD-1472, Section 5.13.7.1 and MIL-STD-454.

3.4 Documentation

3.4.1 Drawings

All Aero engineering released drawings shall be equivalent to or better than that required by MIL-STD-1000, Category E, Form 3. BCO drawings shall conform to Honeywell commercial standards.

3.4.2 Manuals

Any manuals generated for the SFEP shall be consistent with existing Level 6 manuals and related documentation. SFEP manuals shall supplement Level 6 manuals and shall use identical terminology, style, and referencing procedures so that to the user, the set of manuals form one monolithic set of documentation.

3.4.3 Specifications

Specifications are required for all parts, materials and processes utilized in the fabrication and assembly of this unit. This requirement is necessary to assure the validity of Qualification Test Results.

3.4.4 Test Plans

Test plans shall be per Aero Design Procedures paragraph 5.3.

3.5 Logistics

Maintenance procedures, supply, facilities and facility equipment, requirements shall be per the individual SFEP procurement specification.

3.6 Personnel and Training

The Personnel and Training requirements relating to the SFEP shall be per individual procurement specification.

3.7 Major Component Characteristics

3.7.1 Hardware Components

3.7.1.1 Bus

The bus provides a common communication path between units of the SFEP system. The bus is asynchronous in design, permitting units of varying speeds to operate efficiently on the same system. The design of the bus permits the following types of communications to exist:

- Memory Transfers
- Interrupts
- Command Transfers

The maximum system configuration supported by a single bus is 23 connectable units. In this usage, a connectable unit is one which interfaces the bus and generally occupies one connector slot. The number of I/O devices on a single bus may be greater than this number because many of the units support several I/O devices through the same connectable unit. For very large systems, buses may be interconnected via the Inter System Link (ISL) described in paragraph 3.7.1.7.

The bus permits any two units to communicate with each other at a given time via a common (shared) signal path. Any unit wishing to communicate, requests a bus cycle. When that bus cycle is granted, that unit becomes the Master and may address any other unit in the system as the Slave. All transfers are in the direction of Master to Slave only. Some types of bus interchange require a response (Read memory for example). In cases where a response is required, the requestor assumes the role of the Master, indicates that a response is required, and

3.7.1.1 Bus (Continued)

identifies itself to the Slave. When the required information becomes available (depending on Slave response time), the Slave now assumes the role of the Master, and initiates a transfer to the requesting unit.

This completes the interchange which has taken two bus cycles in this case. Intervening time on the bus between these two cycles may be used for other system traffic not involving these two units.

A distributed Tiebreaking Network provides the function of granting bus cycles and resolving simultaneous requests. Priority is granted on the basis of physical position, top priority being given to the first unit on the bus. The logic to accomplish the tiebreaking function is distributed among all units on the bus. In any system, memory is granted highest priority and the CPU the lowest with other units being positioned on the basis of their performance requirements.

In security discussions, references are made to the virtual bus and the absolute bus and drawings occasionally show two separate buses. There is one physical bus and the term virtual or absolute refers only to the type of address that is on the bus at any given time. Since transfers between some units on the bus are always virtual (CPU to memory) and others are always absolute (SPM to memory), it is illustrative to show two buses in drawings.

3.7.1.2 Central Processing Unit (CPU)

The Level 6/40 CPU was selected for the SFEP application because it met the SFEP performance requirements, was securable, and was at a stage of development whereby the SPM interface requirements could still influence the design.

3.7.1.2.1 Standard Level 6/40

The Level 6/40 CPU has a contemporary architecture with some of the key features being:

- 18 program visible general registers including multiple accumulators, multiple address, index and control registers.
- Bit, byte, and word instructions.
- Bit test, set, and mask capability.
- Immediate register to register and register to memory operation.
- 64 interrupt levels.
- Multiple vectored trap structure
- Trap support of interpretive implementation of features like floating point functionality.
- Hardware supported context save and restore.
- Multiple addressing modes including indexing, indirect, base plus displacement, program counter relative auto increment/decrement, etc.

3.7.1.2.1 Standard Level 6/40 (Continued)

- Memory management option
- Scientific processing unit
- Business processing unit
- Stack, queue management

3.7.1.2.2 Level 6/40 Security Modification

To meet security requirements, both hardware and firmware in the Level 6/40 CPU may have to be modified. As a design objective, the changes to the CPU hardware shall be minimized and changes shall be implemented in the firmware whenever possible.

The CPU shall be modified to: provide three (3) special instructions (CALL, RETURN, VALIDATE); change privilege for two (2) instructions (RETURN FROM TRAP, I/O); provide for recoverability from SPM traps; and provide process local trap save areas.

The CALL instruction is used to transfer to an inner ring procedure to accomplish more privileged operations than those allowed at the current ring. The instruction shall do the following. If the SPM access checks are passed, then: the effective address (EA) (e.g., from a previously defined base register) shall be forced into the program counter (P) register; the old ring number shall be made available to software (e.g., put it into a CPU data register); and the new ring number shall

be forced into the system status (S) register. If the SPM access checks are not passed, a trap shall be generated by the SPM.

The RETURN instruction is used to return from an inner ring procedure back to the outer ring from which the call originated. The instruction shall do the following. If the SPM access checks are passed, then: the effective address shall be forced into the program counter; and the specified ring number (e.g., from a CPU data register) shall be forced into the system status register. If the SPM access checks are not passed, a trap shall be generated by the SPM.

The VALIDATE instruction is used to perform argument validation in support of the CALL/RETURN.mechanism. The instruction shall make known to the software the access privileges of a specified memory area to a specified ring.

The privilege level of the RETURN FROM TRAP instruction shall be changed from unprivileged to privileged.

All I/O instructions shall be changed from privileged to unprivileged to allow non-kernel software to do I/O.

3.7.1.2.2 Level 6/40 Security Modification (Continued)

To allow recovery from SPM traps, the trap handling firmware shall be modified to save the SPM fault registers in the Trap Save Area (TSA). If it is more convenient to do so, this feature may be implemented in the SPM.

Process local TSA shall be provided by modifying the trap handling firmware to add an extra level of indirection on the Next Available Trap Save Area Pointer (NATSAP).

Performance requirements may dictate hardware support of stack pointer. If required, this function shall be mechanized to save and restore the stack pointer base register on the following: traps; interrupts; and LEVEL, CALL, RETURN, and RETURN FROM TRAP instructions.

3.7.1.3 Security Protection Module (SPM)

The function of an SPM is to mediate, through a descriptor structure, all interactions between elements of a protected minicomputer.

Each SPM shall contain the following functions:

a. The current and effective ring number of each requestor it services;

b. a pointer (Descriptor Base Root) to the set of descriptors which describe the accessible resources for each requestor;

c. a mechanism by which the protection state and set of resource descriptors may be initialized for each requestor - this mechanism is generally under the control of the associated processor;

d. a mechanism by which the SPM may walk through the descriptor structure to locate the proper descriptor applying to a requested resource;

e. a mechanism by which the SPM may evaluate the propriety of a requested access based on the following information: the identity of the requestor, the access mode of the request, the resource requested, the current protection state of the SPM for the requestor, and the requestor's descriptor for the resource;

f. a mechanism by which the protection state of a requestor may be changed, in a well-defined manner; and

g. a Fast Access Descriptor Store (FADS) in which the SPM may place fast access copies of recently referenced descriptors.

3.7.1.3 Security Protection Module (SPM) (Continued)

h. A mechanism for mapping virtual resource references to real resource references.

These SPM functions shall support the following system level functions as specified in the following subparagraphs.

- Process Initiation
- Memory Access
- I/O Access
- SPM Access
- CALL/RETURN/VALIDATE
- SPM Initiated Traps
- Traps and Interrupts

3.7.1.3.1 Process Initiation

The SPM will be initialized for a new process via a CPU dispatch function sending the SPM the absolute address of the new Descriptor Base Root (DBR). The SPM shall initate the dispatch sequence only if the issuing processor is operating in the kernel domain, the device address indicated by the I/O order is the absolute channel address, and the function code indicates dispatch. The SPM shall immediately block the CPU upon recognition of the dispatch command. The SPM shall access memory to fetch the new DBR starting at the absolute memory address indicated on the data bus, and retain this DBR within its internal memory until another dispatch order is issued. The DBR information shall contain the new memory DBR and I/O DBR. The SPM shall then mark as invalid all memory descriptors contained within its Fast Access Descriptor Store (FADS). The SPM shall not invalidate memory descriptors

1

for I/O devices in process (i.e., active devices - mapped). At the completion of these steps, the SPM shall unblock the CPU and allow normal operation. The security kernel shall insure that process-local information contained in CPU registers is not passed to the new process during dispatch.

The Descriptor Base Root (DBR) establishes the set of descriptors for a process. The DBR provides the mechanism for the SPM to find the memory and I/O descriptor trees for resource access mediation for a process. The DBR construct shall support a structure which defines distinct name spaces for I/O devices and memory. Thus a name designated as an I/O device name will be interpreted in the I/O device descriptor name space; a name designated as a memory address will be interpreted in the memory descriptor name space. This structure requires the DBR to have two components: the first describes the set of memory descriptors, the second describes the set of I/O descriptors.

Each component of the DBR shall contain the following information:

BASE - The Base field shall specify the physical memory address of the tree of descriptors describing the resources of the process. This field shall support as a minimum identical precision as the corresponding field in indirect memory descriptors (i.e., 16 bits).

LIMIT - The LIMIT field shall define the size of the described resource. The LIMIT field shall have sufficient size to allow resolution to a single descriptor within a segment.

TYPE - The TYPE field shall identify the type of the DBR component. The TYPE field shall support a minimum of two

3.7.1.3.1 Process Initiation (Continued)

encodings. One encoding shall identify a direct component DBR which describes an array of resource descriptors contained in a segment; another encoding shall identify an indirect component DBR which describes an array of descriptors that describe pages containing arrays of resource descriptors.

3.7.1.3.2 Memory Access

The SPM shall mediate all references to memory made either by a CPU or a mapped I/O device. When the memory reference is made, the memory address presented on the bus shall be intercepted by the SPM associated with that process or I/O device. The SPM translates this virtual address into a physical memory address. The physical address is then presented to memory, and the appropriate read or write access is made. The data going to and from memory shall not be examined by the SPM.

The following paragraphs present the specifications for the memory descriptors, the multilevel descriptor structure, the memory access sequence, and the memory access control rules for memory reference.

3.7.1.3.2.1 Memory Descriptor

Memory descriptors provide the data for the SPM to convert virtual memory addresses presented by requestors (i.e., CPU's or mapped I/O devices) to absolute memory addresses. In addition, the memory descriptors provide the access control data for the SPM to verify that the process has the required access privilege to the referenced memory location. The information contained in a memory descriptor and the interpretation of the descriptor fields by the SPM is specified below:

3.7.1.3.2.1 Memory Descriptor (Continued)

DT - Directed Trap. The DT field of the descriptor provides for software directed hardware traps. One encoding is a no fault condition; all other encodings shall cause the SPM to fault. A minimum of four encodings shall be supported.

R1, R2, R3 - Ring brackets. The ring brackets are used to restrict the process to certain types of access when executing in a given ring. For access rules, see 3.7.1.3.2.4; for ring definition see 3.1.1.1.2. It must be true that R1≤R2≤R3. The term write bracket shall apply to rings 0 to R1 inclusive. The term read bracket shall apply to rings 0 to R2 inclusive. The term execute bracket shall apply to rings R1 to R2 inclusive. The term call bracket shall apply to rings R1 to R3 inclusive. Each ring bracket field specifying a privilege ring (R1, R2 or R3) shall support a minimum of 4 encodings.

R, W, E - Read, Write, Execute permission. These fields define allowed modes of access to the described resource. Each field shall have two values (ON and OFF); if ON, the respective mode of access shall be allowed, if OFF the respective mode of access shall be denied. Read permission refers to a data or address constant fetch from memory; Write is a store into memory; and, Execute is an instruction fetch from memory.

3.7.1.3. Memory Descriptor (Continued)
2.1

A - Access Field. The access field shall determine whether the access control fields (i.e., ring brackets and R, W, E permission fields) of the descriptor are to be used to control access to all resources described by the descriptor regardless of the number of subsequent levels of address translation. If the A field is ON, then this descriptor's access control fields shall apply; if OFF, either an inferior or superior descriptor must provide the necessary access control.

T - Type field. The type field shall identify the type of the descriptor. One encoding shall identify a direct memory descriptor which directly describes a memory segment. One encoding shall identify a direct memory descriptor which directly describes a memory page. One encoding shall identify an indirect descriptor which describes an array of inferior descriptors. A minimum of three encodings shall be supported.

BASE - The BASE field shall specify the physical memory address of the base of the memory element described. This field must have sufficient precision to address memory resources without waste of physical address space. Thus, direct memory descriptors shall support a BASE field allowing, as a minimum, specification to the physical page address (modulo 128 or 13 bits). Indirect memory descriptors shall support a BASE field providing a factor of eight increase in resolution over that provided by the minimum direct descriptor resolution specified above; (i.e., modulo 16 or 16-bits).

3.7.1.3. Memory Descriptor (Continued)
2.1

LIMIT - The LIMIT field shall specify the size of the defined resource. The LIMIT field shall have sufficient size to allow resolution to a single memory location within a segment, (i.e., 11 bits).

U, M - Used, Modified field. The U, M fields shall record and limit the usage of the described resource. The U field shall have two values (ON and OFF); if OFF and the resource is accessed in any mode, the SPM shall update the value to ON. The M field shall have two values (ON and OFF); if OFF and the resource is accessed in the write mode, the SPM shall update the value to ON. The U, M field is specified to support general memory management.

IOCT - I/O Count. The IOCT field of direct memory descriptors shall be incremented by the SPM at each initiation of an I/O operation in/out of the described resource. This field shall be used by system software to determine the existence of I/O operations in progress within a resource in order to maintain the resource in memory until all outstanding I/O has completed. A minimum count of 16 shall be supported.

The interpretation of the above memory descriptor fields shall be dependent on descriptor level (see 3.7.1.3.2.2). The T, DT, BASE, and LIMIT fields are applicable for each level of memory descriptor. The IOCT, U, and M fields are referenced and updated only for direct descriptors. The access control fields R1, R2, R3, R, E, and W are only applicable for the descriptor which has its A field ON.

3.7.1.3. Multilevel Memory Descriptor Structure
2.2

SFEP supports a general three-level memory descriptor system allowing for the implementation of segments, pages, and paged descriptor segments. The first descriptor table can be considered to be the page table of the descriptor segment, the second table is a page of the descriptor segment, and the third table is the page table for the segment. The indirect descriptors in the descriptor segment are called segment descriptors and the direct descriptors in the page tables are called page descriptors.

The virtual memory address field presented by a processor is shown in Figure 7. The field is 24 bits, however, the virtual address is restricted to 2^{20} words or 2^{21} bytes. A word consists of 2 bytes and a byte is an 8-bit information unit. The SPM shall be designed to accept and map a virtual word address of 20 bits. The byte bit shall be passed unmodified by the SPM. The most significant bit of the address field shall be the SPM flag bit. If this bit is set (i.e., a "1"), it shall indicate that this is a virtual address and is to be mapped by the SPM. If this bit is reset (i.e., a "0"), it shall indicate that this address is an absolute address and requires no mediation by the SPM.

The 20-bit virtual memory word address presented by a processor shall be considered to consist of four fields, designated a, b, c and d. The translation of a virtual memory address into a physical address is illustrated in Figure 7 and shall proceed as follows:

a. The SPM, given a virtual address, makes its first reference to the first level descriptor table pointed to by the indirect memory descriptor base root (DBR) known to the SPM.

b. The offset into this descriptor table is the first field of the virtual address (a) and the descriptor at that location is referenced.

c. The first level descriptor must be an indirect descriptor. Its pointer is used to access a second descriptor table, and the second part of the virtual address (b) is used as an offset into this second table.

d. If the second level descriptor is indirect, it similarly is used to access a third descriptor table and the third part of the virtual address (c) is used to get the third level descriptor.

e. The third level descriptor must be a direct descriptor. Its pointer is used to find the page of data, and the last part of the virtual address (d) is an offset into the page to obtain the actual data word being referenced.

If second level descriptor is direct, its pointer shall be used to directly access the data segment. The offset into the data segment shall utilize the combined c, d fields to obtain the actual data word being referenced.

In addition to the three-level memory descriptor system specified above, SFEP shall support unpaged descriptor segments. In this case, the memory descriptor portion of the DBR points directly (indicated by DBR T field encoding = DIRECT) to the second level

descriptor table, and the combined a, b field is used to index into this table. If the referenced level 2 descriptor is an indirect descriptor, the Steps (d) and (e) presented above shall be followed. If the referenced second level descriptor is direct, its pointer shall be used to access the data segment. The offset into the data segment shall utilize the combined c, d fields to obtain the actual data word being referenced. Figure 7 also illustrates this alternate memory descriptor structure.

3.7.1.3. Memory Reference Sequence
2.3

As the SPM is walking through the descriptor tree, it shall (at each descriptor level) compare the LIMIT field from the previous level to the offset to be used to access the next level descriptor. If the offset exceeds the associated LIMIT field at any level, the SPM shall trap.

The SPM shall monitor the A-field of each descriptor encountered in its search for a direct descriptor. The access field (ring brackets and R, W, E fields) of the first descriptor encountered with the A-field ON shall define the appropriate access control for the resource. If a direct descriptor is encountered without its A-field or any previous A-field ON, the SPM shall trap.

If the SPM does not encounter a direct descriptor at level 2 or at level 3 of the descriptor structure, the SPM shall trap. A direct descriptor encountered at level 1 shall also result in an SPM trap.

If the SPM encounters a descriptor at any level with its DT field set to a software-directed trap, the SPM shall trap.

Upon encountering the direct descriptor for the accessed memory location, the SPM shall compare its LIMIT field to the page or segment offset specified in the virtual address. If the offset exceeds the LIMIT, the SPM shall trap.

3.7.1.3. Memory Reference Sequence (Continued)
2.3

The SPM shall perform the access checks (i.e., ring brackets and permission fields) obtained from the applicable descriptor (A=ON) as specified in 3.7.1.3.2.4. If any access check fails, the SPM shall trap. To support access checking, the SPM shall compute an effective ring number for each instruction execution as specified in 3.7.1.3.2.4.

The SPM shall set the U bit of the direct descriptor in memory describing the accessed resource ON, if off, regardless of access mode. In addition, the SPM shall set the M bit of the direct descriptor in memory describing the accessed resource ON, if off, if the write mode of access is indicated.

Upon performing all checks, the SPM places the physical memory address of the accessed resource on the address bus (i.e., physical address = BASE field of direct descriptor + offset field of virtual address). The memory shall respond appropriately via the data bus.

For performance reasons, the SPM shall retain copies of recently referenced direct descriptors in its Fast Access Descriptor Store. The SPM shall always attempt to locate a descriptor for the accessed resource within its FADS prior to accessing the memory descriptor tree in memory.

3.7.1.3. Memory Access Rules
2.4

The following rules shall specify the required interpretation of the access control information (i.e., ring brackets and R, W, E permission fields) obtained from the applicable memory descriptor (A=ON). The effective ring number (Reff) is the maximum of the ring numbers in which the process is operating (R_{cur}) and all R1 fields encountered in all descriptors during address preparation. The value of Reff shall be initialized to R_{cur} at the beginning of each instruction cycle and shall apply to the instruction fetch and all references until the next instruction fetch. For each descriptor encountered between instruction fetch and operand fetch, a new value of Reff shall be computed as the maximum of the current Reff and R1 in the descriptor and this new Reff shall apply to the fetch of subsequent indirect addresses or data.

a. Write permission if and only if (W = ON) and Reff $\leq$ R1)

b. Read permission if and only if (R=ON) and (Reff $\leq$ R2).

c. Execute permission if and only if (E=ON) and (Reff $\leq$ R2).

d. The use of R3 and the precise rules for entry/return to/ from a procedure resource are specified in 3.1.2.3.2.2.1. In general, call permission if and only if (E = ON) and (Reff $\leq$ R3).

3.7.1.3.3 I/O Access

Two types of I/O will be supported by the SFEP. These are Direct Memory Access (DMA) and Programmed I/O (PIO). The DMA devices, once initiated by the processor, reference memory independent of the processor to perform the required data transfer. PIO does not reference memory, all data transfer takes place between the device and the processor. Included in PIO are status requests to the DMA devices. Two classes of DMA devices shall be supported, these are the mapped and premapped devices. The SPM shall mediate all references from the mapped devices to memory, while the premapped devices can reference memory directly, i.e., without SPM mediation. The SPM shall mediate all accesses from the processor to the device.

The following paragraphs present the specifications for the I/O descriptors, the I/O descriptor structure, the I/O access sequence, and the I/O access rules for I/O reference.

3.7.1.3.3.1 I/O Descriptors

I/O descriptors provide the mechanism for the SPM to convert virtual device addresses presented by requestors (i.e., CPU's) to absolute device addresses. In addition, the I/O descriptors provide the access control mechanism for the SPM to verify that the process has the required access privilege to the referenced device.

The information contained in an I/O descriptor and the interpretation of the descriptor fields by the SPM is specified below.

3.7.1.3. I/O Descriptors (Continued)
3.1

DT - Directed Trap. The DT field of the descriptor provides for software directed hardware traps. One encoding is a no fault condition; all other encodings shall cause the SPM to fault. A minimum of four encodings shall be supported.

R1, R2, R3 - Ring brackets. The ring brackets are used to restrict the process to certain types of access when executing in a given ring. For access rules, see 3.7.1.3.3.4 for ring definition, see 3.1.1.1.2. It must be true that R1 $\leq$ R2; it is not required that R3 $\geq$ R2. Each ring field shall support a minimum of four encodings.

R, W, E - Read, Write, Execute permission. These fields define allowed modes of access to the described resource. Each field shall have two values (ON and OFF); if ON, the respective mode of access shall be allowed, if OFF the respective mode of access shall be denied. Read permission refers to a transfer from the device to memory; Write refers to a transfer from memory to the device; and Execute refers to trusted device control operations.

A - Access field. The access field determines whether the access control fields (i.e., ring brackets and R, W, E permission fields) of the descriptor are to be used to control access to all resources described by the descriptor regardless of the number of subsequent levels of address translation. If the A field is ON, then this descriptor's access control fields shall apply; if OFF, either an inferior or superior descriptor must provide the necessary access control.

MT - Mapping Type. The MT field shall specify whether the described device is mapped or pre-mapped. The MT field shall have two values (ON and OFF); if ON, then the described device is a pre-mapped I/O device; if OFF, then the described device is a mapped I/O device.

T - Type field. The type field shall identify the type of the descriptor. One encoding shall identify a direct I/O descriptor which directly describes an I/O device. One encoding shall identify an indirect descriptor which describes an array of inferior I/O descriptors. A minimum of two encodings shall be supported.

3.7.1.3. I/O Descriptors (Continued)
3.1

BASE - The BASE field shall specify the physical memory address of the base of a set of direct I/O descriptors. The BASE field is only applicable for indirect I/O descriptors. The BASE field shall have identical precision as supported by indirect memory descriptors.

DEVICE - The DEVICE field shall identify the physical device name (i.e., channel number) to be used upon access to this descriptor. The DEVICE field is only applicable for direct I/O descriptors. The DEVICE field shall have sufficient size and precision to accomodate all device names addressable by the processor (i.e, 10 bits).

LIMIT - The LIMIT field shall specify the number of potentially accessible direct I/O descriptors. The LIMIT field shall have sufficient size to allow resolution to a single direct I/O descriptor within a segment.

U, M - Used, Modified field - The U, M fields shall record and limit the usage of the described resource. The U field shall have two values (ON and OFF); if OFF and the resource is accessed in any mode, the SPM shall update the value to ON. The M field shall have two values (ON and OFF); if OFF and the resource is accessed in the write mode, the SPM shall update the value to ON. The U, M field is specified to support general resource management.

3.7.1.3.1 I/O Descriptors (Continued)

SPM - The SPM designator field shall specify the SPM to be used for mapped I/O mediation (device to memory accesses) in a multiprocessor configuration. The SPM field shall have sufficient size to accommodate all assignable SPM channel members.

CLASS - The CLASS designator field shall specify the device type of the accessed resource to allow mediation of device control operations. The CLASS field shall support a minimum of 16 device types.

The interpretation of the above I/O descriptor fields shall be dependent on descriptor level. The T and DT fields are applicable for each level of I/O descriptor. The BASE and LIMIT fields are applicable to indirect I/O descriptors only. The MT, DEVICE, U, M, SPM, and CLASS fields are referenced only for direct I/O descriptors. The access control fields, R1, R2, R3, R, E, and W are only applicable for the descriptor which has its A field ON.

3.7.1.3.2 I/O Descriptor Structure

SFEP shall support a two level I/O descriptor system allowing for the implementation of paged descriptor segments. The 10-bit virtual device address (i.e., channel number) shall be considered to consist of two fields, designated e, f. The translation of a virtual device address into a physical address as illustrated in Figure 8 shall proceed as follows:

3.7.1.3. I/O Descriptor Structure (Continued)
3.2

(a) The SPM, given a virtual address, makes its first reference to the first level descriptor table pointed to by the indirect I/O descriptor base address known to the SPM via the DBR received during process dispatch.

(b) The offset into this descriptor table is the first field of the virtual address (e) and the descriptor at that location is referenced.

(c) The first level descriptor must be an indirect descriptor. Its pointer is used to access a second descriptor table, and the second part of the virtual address (f) is used as an offset into this second table.

(d) The second level descriptor must be a direct descriptor. Its pointer (i.e., DEVICE field) is used to access the physical device being referenced.

In addition to the two-level I/O descriptor system specified above, SFEP shall support unpaged descriptor segments. In this case, the I/O descriptor portion of the DBR points directly (indicated by DBR T field encoding = DIRECT) to the second level descriptor table, and the combined e, f field is used to index into this table. The referenced second level descriptor must be direct and shall be used to access the physical device. Figure 8 also illustrates this alternate I/O descriptor structure.

3.7.1.3. I/O Access Sequence
3.3

The SPM shall mediate all accesses made by processors to devices through the I/O descriptor tree structure. The SPM shall utilize the I/O portion of the DBR received during process dispatch to locate the direct descriptor for the device to be accessed. The SPM shall interpret the two fields of the virtual device address as a function of the Type fields encountered in the I/O descriptor tree in memory. At each descriptor level, the SPM shall compare the limit field from the previous level to the offset to be used to access the next level descriptor. If the offset exceeds the associated limit field at any level, the SPM shall trap.

The SPM shall monitor the A-field of each descriptor encountered in its search for a direct descriptor. The access field (ring brackets and R, W, E fields) of the first descriptor encountered with the A-field ON shall define the appropriate access control for the resource. If a direct descriptor is encountered without its A-field or any previous A-field ON, the SPM shall trap.

If the SPM does not encounter a direct descriptor at level 2 of the descriptor structure, the SPM shall trap. A direct descriptor encountered at Level 1 shall also result in an SPM trap.

If the SPM encounteres a descriptor at any level with its DT field set to a software-directed trap, the SPM shall trap.

The SPM shall perform the access checks (i.e., ring brackets and permission fields) obtained from the applicable descriptor (A=ON) as specified in 3.7.1.3.3.4. If any access check fails, the SPM shall trap. To support access checking, the SPM shall compute an effective ring number for each I/O instruction execution as specified in 3.7.1.3.3.4.

The SPM shall set the U bit of the direct I/O descriptor in memory describing the accessed resource ON, if OFF, regardless of access mode. In addition, the SPM shall set the M bit of the direct descriptor in memory describing the accessed resource ON, if OFF, if the write mode of access is indicated.

If a device control operation is indicated by the function code associated with each PIO operation, the SPM shall mediate the operation if initiated by untrusted code. Trusted control operations shall be allowed by the SPM if and only if (E=ON) and (Reff $\leq$ R3) in the access control field for the device. An example of a trusted control operation would be a modify controller microcode operation. If the control operation is initiated by untrusted code, the SPM shall use the function code and the device type (supplied by CLASS field in I/O descriptor) to validate the operation; that is, the SPM shall verify that the operation is valid for that class of device. Since device

control operations may be performed by the CPU sending a control word to the device, the SPM shall be required to examine the data bus in order to perform control operations mediation. Although many device control operations are physically performed by writing a control word to the device controller, certain operations must be allowable with only read permission (R = ON, W = OFF). That is, the SPM shall support the concept of "write ring protection." Thus, control operations shall be categorized into those allowable with only write permission to the device (i.e., W = ON and Reff $\leq$ R1) and those allowable with either read or write permission. For example, both an end file tape and a tape rewind command are implemented by sending a control word to the device controller; however, the end file function must be restricted to processes having device write permission while the rewind function is allowable with either read or write access. As noted previously, these checks are required only for I/O operations initiated by untrusted code. All device control decision tables shall reside solely within the security kernel data base. If an illegal control operation is attempted, the SPM shall trap.

Upon receipt of an I/O order from the processor, the SPM shall block the CPU until the SPM receives an ACK from the mediated device (assuming all SPM checks pass). The SPM shall then release the CPU (i.e., ACK'ed). If a NAK is received by the SPM, the CPU request will be NAK'ed by the SPM.

For PIO devices, no further mediation is required. That is, upon performing all checks, the SPM replaces the virtual device address with the absolute channel number retrieved from the direct I/O descriptor (i.e., Device field), leaving the function code unmodified, and places the I/O command on the address bus. Essentially, the basic SPM I/O mediation only requires:

A. That the device has been assigned to the process, indicated by the presence of an I/O descriptor.

B. That the descriptor defining the I/O device allows access in the requested mode at the effective ring number of the process requesting the transfer.

For performance reasons, the SPM shall retain copies of recently referenced direct I/O descriptors in its Fast Access Descriptor Store. The SPM shall always attempt to locate a descriptor for the accessed resource within its FADS prior to accessing the I/O descriptor tree in memory.

For DMA devices, further SPM mediation is required as specified in the following subparagraphs. The initiating processor shall signal the SPM as to device type (i.e., DMA or PIO). The SPM shall categorize DMA devices as premapped or mapped based on the MT-field in the direct I/O descriptor.

3.7.1.3. Premapped I/O Sequence
3.3.1

The SPM shall determine that a device is to be treated as a premapped I/O device by an examination of the MT bit (i.e., MT=ON) of the direct I/O descriptor. Upon completion of the basic SPM I/O mediation specified above, the following additional functions are required.

The SPM shall obtain a direct memory descriptor for the memory to be accessed via the memory descriptor tree. The normal memory reference sequence specified in 3.7.1.3.2.3 shall be follwed by the SPM, including all specified validity checking. It should be noted that a new Reff must be computed by the SPM for the memory resource. The SPM shall insure that the range of affected memory addresses falls within the range of memory described by the one direct memory descriptor; the SPM shall compare the page or segment offset specified in the virtual memory address plus the specified range to the limit field of the direct memory descriptor. If this comparison fails, the SPM shall trap.

The SPM shall set the U bit of the direct memory descriptor in memory describing the accessed resource ON if OFF; The SPM shall set the M-bit of the direct descriptor in memory ON if OFF, if the write mode (to device) is indicated (i.e., function code must be interpreted by SPM). In addition, the IOCT field of the direct memory descriptor in memory shall be incremented by the SPM to notify the security kernel of the number of I/O operations initiated within the described memory resource.

3.7.1.3. Premapped I/O Sequence (Continued)
3.3.1

For premapped I/O, the SPM shall map the I/O channel number and the starting memory address. The SPM shall receive from the CPU, via two bus cycles, the virtual channel number, the virtual starting address, the range or number of words to be transferred, and a function code indicating mode of access. The SPM shall map the virtual channel number into an absolute channel number using the direct I/O descriptor. The SPM shall map the virtual starting address into an absolute starting address using the direct memory descriptor. The SPM shall pass the range and function code unmodified. Via two bus cycles, the SPM shall send the absolute information to the device. Resultant transfer of data shall occur directly between the device and memory without intervention by the SPM.

3.7.1.3. Mapped I/O Sequence
3.3.2

The SPM shall determine that a device is to be treated as a mapped device by an examination of the MT bit (i.e., MT=OFF) of the direct I/O descriptor. Upon completion of the basic SPM I/O mediation specified above (3.7.1.3.3.3), the following additional functions are required.

The SPM shall obtain a direct memory descriptor for the memory to be accessed (i.e., initial access - unlike premapped I/O, multi-page I/O shall be supported for mapped devices) via the memory descriptor tree. The normal memory reference sequence specified in 3.7.1.3.2.3 shall be followed by the SPM, including all specified validity checking. However, since all resultant memory references by the device are to be mediated individually by the SPM, there is no requirement for access mediation using the direct memory descriptor during mapped device initiation.

3.7.1.3. Mapped I/O Sequence (Continued)
3.3.2

The SPM shall retain (tagged with a unique device identifier) for each active mapped I/O device, the following information. An active device is one for which the SPM has not been notified of I/O termination.

A. The effective ring number at which the device is to access memory. The initially computed value of Reff shall apply for the entire transfer regardless of the number of pages.

B. The memory descriptor tree pointer from the initiating process DBR to allow the SPM to obtain additional memory descriptors if multi-page I/O is performed.

To support multi-bus configurations, the SPM associated with the initiating CPU shall examine the SPM field contained in the I/O descriptor to determine the identity of the SPM responsible for subsequent device to memory access mediation. If the SPM field contains its own device identifier, the SPM continues processing as specified in the subsequent paragraphs. If another SPM is specified, the SPM shall initiate an I/O sequence (using absolute addressing) which transfers the device identifier, the initial memory descriptor, the effective ring number, and memory DBR to the indicated SPM. That SPM shall be responsible for subsequent device-to-memory access mediation. The target SPM shall be that associated with the same bus as the target device.

3.7.1.3. Mapped I/O Sequence (Continued)
3.3.2

The SPM shall set the U bit of the initial direct memory descriptor in memory describing the accessed resource ON if OFF. The SPM shall set the M bit of the direct descriptor in memory ON if OFF, if the write mode (to device) is indicated (i.e., function code must be interpreted by SPM). In addition, the IOCT field of the initial direct memory descriptor shall be incremented by the SPM to notify the security kernel of the number of I/O operations initiated within the described memory resource.

For mapped I/O, the SPM shall map only the I/O channel number. The SPM shall receive from the CPU, via two bus cycles, the virtual channel number, the virtual starting address, the range, and a function code indicating mode of access. The SPM shall map the virtual channel number into an absolute channel number using the I/O descriptor. In order that the proper memory descriptor may be selected, all virtual addresses from the mapped device shall be accompanied by an identification of the requesting device. This shall be accomplished by the SPM by setting the most significant bits of the starting address equal to a unique device identifier and the remaining portion of the starting address to the page or segment offset specified in the virtual starting address. The SPM shall pass the range and function code unmodified. Via two bus cycles, the SPM shall send the modified information to the device.

3.7.1.3. <u>Mapped I/O Sequence</u> (Continued)
3.3.2

When the virtual address associated with each request from the device for data transfer arrives at the target SPM, the SPM shall retrieve the memory descriptor and effective ring number by the device identifier contained in the virtual address. The checking by the SPM during mapped I/O transfer shall be identical to the checking of a memory access by the CPU. The physical memory address placed on the bus by the SPM shall consist of the BASE address of the direct memory descriptor plus the offset provided in the virtual address provided by the device; i.e., the device identifier portion of the virtual address is discarded by the SPM.

If the offset portion of the virtual memory address associated with each request from the device overflows (i.e., exceeds page boundary), the SPM shall retrieve a replacement direct memory descriptor via the stored DBR. The U, M and IOCT fields of the new direct memory descriptor in memory shall be updated appropriately (i.e., as done for the initial descriptor). Permission checking shall then proceed as before. It should be noted that multi-segment I/O need not be supported; only a single segment may be accessed as a result of a single I/O operation.

The SPM (or SPM's in a multiprocessor configuration) shall insure the following condition on mapped I/O operations.

1. The device-to-memory access mediating SPM shall be that specified in the device descriptor.

3.7.1.3.3.4 I/O Access Rules

The following rules shall specify the required interpretation of the access control information (i.e., ring brackets and R, W, E permission fields) obtained from the applicable I/O descriptor (A=ON). The rules for determining the effective ring number (Reff) for the process are as specified in 3.7.1.3.2.4 for memory access.

a. Write permission if and only if (W=ON) and (Reff ≤ R1).

b. Read permission if and only if (R=ON) and (Reff ≤ R2).

c. Control permission if and only if (E=ON) and Reff ≤ R3). If control permission is granted, the requesting process shall be assumed to be trusted and, thus, no function code checking as specified in 3.7.1.3.3.3 shall be performed.

3.7.1.3.4 SPM Access

Direct SPM access shall be allowed by privileged CPU functions. These include the DISPATCH, SELECTIVE DESCRIPTOR INVALIDATION, and SPM T&D. The DISPATCH function is specified in 3.7.1.3.1. The other two functions shall be implemented as specified below.

3.7.1.3.4.1 SELECTIVE DESCRIPTOR INVALIDATION

The SPM shall respond to DESCRIPTOR INVALIDATION orders from the CPU by retrieving the specified descriptor from memory and overwriting the invalidated descriptor in FADS.

3.7.1.3. SELECTIVE DESCRIPTOR INVALIDATION (Continued)
4.1

The kernel shall be capable of invalidating all memory segment descriptors, selected memory segment descriptors, all memory page descriptors, selected memory page descriptors, selected I/O memory descriptors, and all I/O device descriptors. An I/O output command, with appropriate function code, issued by the kernel to the appropriate SPM (absolute device address) shall be used to perform selective descriptor invalidation. With the exception of I/O memory descriptors, all descriptor invalidation orders issued by a processor are to be directed to its associated SPM. In support of multibus mapped I/O devices, the initiating processor shall direct the selective I/O memory descriptor invalidation command directly to the mediating SPM (device to memory accesses).

If the absolute channel provided by an I/O output command equals the SPM and the CPU is operating in the kernel domain, and the function code indicates memory segment descriptor invalidation, the SPM shall mark all memory page descriptors invalid. If the function code indicates selective memory segment descriptor invalidation, the SPM shall mark the direct segment descriptor for the segment identified on the data bus invalid. If the function code indicates selective memory page descriptor invalidation, the SPM shall mark the direct page descriptor for the page within the segment identified on the data bus invalid. In none of the above cases shall the SPM invalidate memory descriptors for active I/O devices. If the function code indicates I/O descriptor invalidation, the SPM shall mark all I/O descriptors invalid. If the function code indicates I/O memory descriptor invalidation, the SPM shall mark the memory descriptor for the absolute I/O device identified on the data bus invalid.

3.7.1.3. <u>SPM T&D</u>
4.2

The SPM shall be directly accessible to privileged SPM T&D orders.

3.7.1.3. <u>CALL/RETURN/VALIDATE</u>
5

The SPM shall support the CALL, RETURN, and VALIDATE instructions. These instructions are used to assure that all ring crossings meet the isolation requirement. The following paragraphs present the access rule specifications for these instructions. The memory reference descriptor structure shall support these instructions.

The SPM shall mediate the virtual entry point address placed on the address bus by a process executing the CALL instruction via the memory descriptor tree structure as for a normal memory reference. However, unlike normal memory references, no mapping is performed. That is, the SPM shall only validate the virtual entry point address (i.e., caller has execute access) and compute a new value of R_{cur}.

If the effective ring number of the caller is outside the call bracket, the SPM shall trap. Similarly in order to control entry to the called procedure, the SPM shall insure that only location zero of the called procedure (defined by the direct page or segment descriptor) be a valid entry point. If a call to an invalid entry point is attempted, the SPM shall trap. It should be noted that the entry point limitation is the degenerate example of a call limiter restriction specifying the maximum offset within a segment or page to which a call can transfer to be zero.

The mechanism that accomplishes process requested cross-ring movement shall be implemented as follows. An inner ring procedure that is callable from an outer ring is defined as a "gate" by specifying in

the ring brackets of the descriptor for the procedure segment a value of R3 that is different from R2. Normally, transfers to a segment cannot be made from rings above R2. However, a call instruction is allowed to a procedure if the call is made from a ring less than or equal to R3. If such a call is made, the new value of R_{cur} becomes R2, and execution continues. The value of Reff after address preparation for the call instruction is used in the comparison with R2 and R3. The tests made in the call are as follows:

Reff > R3	entry denied, trap (outside call brackets)
R2 < Reff ≤ R3	entry allowed, R2 becomes R_{cur}
Reff ≤ R2	entry allowed, R_{cur} unchanged

The checks on call shall not preclude using the call instruction to transfer to a procedure from within its execute bracket. Nor shall it be required that a segment be a gate (i.e., R2 < R3) in order to be called from within its execute bracket. Thus, the call bracket is defined as R1 to R3, with R2 being the new ring of execution if the segment is a gate and the call is from outside R2.

The only requirements for the cross-ring return instruction are that the returning procedure be able to specify the ring to which to return and that returns to inner rings be prohibited. Otherwise with return shall operate like a transfer instruction. Assume that Rto is the ring to which the procedure desires to return:

Reff ≤ Rto	Rto becomes R_{cur}
Reff > Rto	return denied, trap (inward return)

On a VALIDATE instruction, the SPM shall return to the CPU access rights to a memory area for a specified ring.

3.7.1.3. SPM Generated Traps

6

The SPM shall generate a variety of traps defined by the checking conditions within the SPM. At least one trap type (defined by a trap vector/save area) shall be reserved for SPM generated security fault conditions. As part of the saved state of the process, the SPM shall store suitable information in memory to allow the security kernel to determine the cause of the trap and allow suitable recovery to proceed. A list of the conditions that shall cause the SPM to initiate the trap condition is presented below:

a) Memory Access

1. Directed fault
2. Limit violation
3. Write violation: (Reff $>$ R1) or (W = OFF)
4. Read violation: (Reff $>$ R2) or (R = OFF)
5. Execute violation: (Reff $>$ R2) or (E = OFF)
6. Missing access field: (A = OFF at all levels)*
7. No direct descriptor: (T = INDIRECT at all levels)*
8. Level 1 direct descriptor encountered *

b) CALL/RETURN

1. CALL violation: (Reff $>$ R3)
2. Entry violation: (OFFSET $\neq$ 0)
3. Return violation: (Reff $>$ Rto)

---

*Since descriptor trees are generated only by verified kernel software, the occurrence of these conditions would indicate a hardware failure, rather than a security violation.

3.7.1.3. <u>SPM Generated Traps</u> (Continued)
6

c) <u>Device Access</u>

1. Directed fault
2. Limit violation (indirect descriptors only)*
3. No direct descriptor; (T =indirect at all levels).*
4. Level 1 direct descriptor encountered*
5. Missing access field; (A=OFF at all levels)*
6. Write violation; (Reff > R1) or (W-OFF)
7. Read violation; (Reff > R2) or (R=OFF)
8. Control violation; (Reff > R3) or (E=OFF)

The SPM shall store security trap information in the appropriate trap save area in memory upon trap occurrences for access by the security kernel trap handler.

---

*Since descriptor trees are generated only by verified kernel software, the occurrence of these conditions would indicate a hardware failure, rather than a security violation.

3.7.1.3. Traps and Interrupts
7

The SPM shall force R_{cur} to 0 on every occurrence of a trap or interrupt to allow CPU firmware to access to the appropriate save/restore areas in kernel space. It shall also allow the CPU generated virtual addresses for trap/interrupt handler entry points to be accessed at kernel level of privilege.

3.7.1.4 Multi-Line Communications Processor (MLCP)

The MLCP supports up to 8 full duplex (FDX) lines on a single board using a single bus interface slot.

Line dependent logic is contained on Communications Line Adapter (CLA) subboards (called daughter boards). Each CLA consists of either one or two line interfaces depending on the complexity of the particular line adapter functions and data set interface to be supported, and upon the availability of LSI for the interface. These CLA's are interchangeable, and up to four CLA's on each MLCP are provided as shown in the functional block diagram, Figure 11.

Each line on an MLCP is considered as a FDX data path, and each line direction is a channel to the SFEP system. Each channel is capable of either sending or receiving a communications type of data stream between memory blocks and the communications interface via DMA type of operation. In the process of transferring this data stream, the MLCP is capable of fully delimiting the data with special character generation, detection, and block check information. The MLCP is also capable of edit and conversion of prespecified sequences in the data stream. Control of these data stream functions is designated by configuration of a Communications Parameter Table (CPT) and a Communications Control Table (CCT) which can be loaded by software into the MLCP.

3.7.1.4.1 Communication Line Adapters (CLA)

The MLCP supports up to 4 line adapters each of which contains one or two interfaces. The CLA characteristics are as follows:

3.7.1.4.1.1 Synchronous Line Adapter

- 2 lines FDX
- Speeds up to 10,800 bits per second
- EIA RS 232 type of interface
- Support for Basic Mode ASCII and BSC type of communications line protocol

3.7.1.4.1.2 Asynchronous Line Adapter

- 2 lines FDX
- Speeds up to 9,600 bits per second
- EIA RS 232 type of interface
- 1 or 2 stop bits
- Selection of speeds by parameter

3.7.1.4.1.3 Modem Bypass Synchronous Line Adapter

- 2 lines FDX
- Capable of selecting a synchronous set of speeds by program setable parameters
- EIA RS 232 type of interface
- Support for Basic Mode ASCII and BSC type of communications line protocol

3.7.1.4.1.4 MIL-188 Line Adapter

- 1 line FDX
- Speeds up to 10,800 bits per second
- MIL-188 type of interface
- Support for Basic Mode ASCII and BSC type of communications line protocol

3.7.1.4.1.5 Programmable Asynchronous Line Adapter

- 1 line FDX
- Speeds up to 10,800 bits per second
- EIA RS 232 type of interface
- 1 or 2 stop bits
- Program setup of speed to 1/2 bit accuracy.

3.7.1.4.1.6 HDLC Line Adapter

- 1 line FDX
- Speeds up to 72K bits per second
- EIA RS 232 and Broadband CCITT V35 type of interface
- Support of HDLC line protocol.

3.7.1.4.1.7 Broadband Line Adapter

- 1 line FDX
- Speeds up to 72K bits per second
- Broadband CCITT V35 type of interface
- Support for Basic Mode ASCII and BSC type of communications procedures.

3.7.1.4.1.8 ACU Line Adapter

- Interface 801C type of automatic calling unit
- 2 lines FDX.

3.7.1.4.2 Verification

The MLCP receives a starting address and a range or tally count via a CPU IOLD instruction. The starting address will be a 24 bit absolute address in the case of premapped I/O or a 1 bit SPM flag, 10 bit Device Id and 12 bit offset fields in the case of mapped I/O. Although the operating mode is selectable via the MT bit of the I/O descriptor, the MLCP will normally run in the mapped I/O mode.

3.7.1.4.2.1 Premapped I/O

In premapped I/O, it is verified that the device, the affected memory locations and the access mode are all valid for the initiating process during setup. The DMA action then takes place without further checking. This results in the need for verification that as a minimum the MLCP firmware and/or hardware cannot modify the:

A. Starting address

B. Range count

C. Access mode

D. Message contents

3.7.1.4.2.2 Mapped I/O

In mapped I/O, it is verified that the device and access mode are valid for the initiating process. Each reference from the device to memory is then checked and mapped. This reduces the verification task, but as a minimum it must still be verified that the MLCP firmware and/or hardware cannot modify the:

A. Starting address device Id or SPM flag bit

B. Message contents

3.7.1.5 Memory

Memory is ultimately planned to be available in several variations relative to speed and modularity. Memories of varying speeds and modularities are usable on the same system.

The memory module presently available is a semiconductor memory that contains up to 32K words in blocks of 8K. This module will occupy one bus slot and multiple modules can be attached to a single bus. This module will have an access time of 600 nanoseconds.

The memory module is available with either parity or EDAC (Error Detection and Correction) protection. EDAC will detect and correct single bit failures and detect multiple bit failures.

3.7.1.6 6000/Series 60 Interface Unit (IU)

The IU provides a communication path between the SFEP bus and the host computer as shown in Figure 1. The host connection is through the Direct Channel Adapter (DCA) of the 6000/I/O Multiplexer.

The IU operates in a "scatter-gather" mode via a list processing technique that provides software generality and efficiency.

There are two modes of data formatting that result because of the 16 bit data bus in the SFEP and the 36 bit data bus in the DCA. These two modes, ASCII and binary, are software selectable and the formats are shown in Figure 12. Details of the IU operation can be found in ESD-TR-76-355.

3.7.1.7 Inter System Link (ISL)

The ISL may be used to interconnect two NML buses. One or more ISL's can connect to a single bus so that multi-bus systems may be interconnected together. Figure 13 illustrates a multi-bus system with its ISL's.

An ISL consists of two NML boards interconnected by a cable which may be up to 25 feet long. Each of the halves of an ISL, (called ISL twins) are identical. The ISL may be used to pass memory references, I/O commands or interrupts from one bus to the other. Each ISL twin can, therefore, assume the bus visibility of a memory, an I/O controller or a Processor at different times as it intercepts a bus transfer on one bus and reinitiates it on a different bus. At system configuration time, each ISL twin is made cognizant of certain memory addresses and certain channel numbers to which it should respond. During system operation, each ISL twin monitors all bus traffic and responds to individual bus cycles within its range on behalf of the actual unit to which the cycle was directed. The ISL twin passes the information to its remote twin which reinitiates the bus cycle. The response cycle from the unit, if any, follows the same route in the reverse direction and is finally routed to the originating unit.

3.7.1.7 Inter System Link (ISL) (Continued)

The ISL twins are buffered and operate asynchronously from each other. The intent of the design is to achieve maximum bus performance for traffic which remains within the bounds of a single bus. Traffic passing through the ISL to the remote bus will, of course, be somewhat delayed. For example, read from a memory on the same bus which takes 1 microsecond might take 1.5 microseconds through the ISL to an adjacent bus.

The ISL will respond to a memory reference which has an address that the ISL has been preprogrammed to accept. The ISL has a memory mask register, see Figure 14, in which a 1 in a cell represents a 8192 word block of memory and any memory reference with an address in the 8192 word block will be accepted by the ISL. The ISL then adds a signed, 10 bit, displacement factor δ_1 which translates the address for the memory on the other bus. Both the Mask Register and the displacement factor are software loadable.

IOLD instructions to the ISL work similarly. There is a channel number mask register and a displacement factor. The result is any CPU can initiate a transfer from a memory on any bus to a controller on any bus as long as the ISL's have been properly programmed.

3.7.1.7 Inter System Link (ISL) (Continued)

The displacement factor is a 10 bit number aligned with the 10 most significant (0-9) bits of the address bus. In a system with an SPM, the ISL must not be allowed to modify bit 0 because it would result in the absolution or virtualization of an address passing through it. Therefore, the software associated with an ISL must be verified.

In a system where the ISL is used to extend the bus (no CPU/SPM on the remote bus), the ISL will be programmed to accept virtual addresses and the displacement factor will be zero. In a multiprocessor environment where the ISL is used to interconnect buses each containing CPU/SPM's, the ISL must be programmed to not respond to virtual addresses because of the ambiguity of the source when it arrives at the SPM on the remote bus.

The following rules apply in a multiprocessor environment:

1. There is an SPM for each CPU.
2. In a system utilizing mapped I/O, there can be no more than one CPU/SPM pair per bus.
3. The ISL shall not respond to virtual memory references.
4. The SPM on the bus containing the I/O device must map the device.

3.7.1.7 Inter System Link (ISL) (Continued)

Since any CPU/SPM can initiate a transfer to/from any device but only the SPM on the bus with the device can map it, the initiating SPM must transfer the appropriate memory descriptor, DBR and Reff to the SPM that will do the mapping.

3.7.2 Software Components

The Secure Front-End Processor software is composed of the major components presented in Section 3.1.1.2. The following subparagraphs will specify the characteristics of these components required to satisfy SFEP functional requirements.

3.7.2.1 General Issues

The following subparagraphs specify general philosophy to be followed in the design of the SFEP software components.

3.7.2.1.1 SFEP Software Design Philosophy

The major SFEP software components shall be structured with respect to application dependency. That is, as a design objective, the security kernel shall be suitable for all applications (i.e., front-end processor and stand-alone network applications) without modification. Similarly as a design objective, the SFEP operating system shall be expandable to allow satisfaction of all application requirements. The basic SFEP operating system components shall be structured such that additional functionality may be included without modifying overall operating system design. The SFEP communications subsystem shall be the only software component which is completely applications dependent.

3.7.2.1.2 Distributed Versus Separate Processes

Following MULTICS, SFEP shall utilize the concept of distributed processes. That is, the SFEP supervisor (i.e., kernel and operating system) will not operate in a dedicated process or address space. Instead, the supervisor is distributed --- its procedure and data segments are to be shared among all SFEP processes. The execution of the supervisor in the address space of each process facilitates communication between user procedures and supervisor procedures and allows the simultaneous execution of supervisory functions by several processes. If separate processes were utilized, additional overhead would result from process identification and request queueing for service. The apparent advantage of separate processes is isolation; however, the necessary isolation within a distributed process is provided by the ring mechanism. That is, the necessary isolation is achieved by having the supervisory and user procedures execute in separate domains (i.e., rings). Thus, the distributed process philosophy shall be followed in SFEP software design.

3.7.2.1.3 Security Versus Policy

In any system, it is desirable to separate policy and mechanism. This is particularly true in secure systems, where the size and complexity of the kernel must be minimized. The SFEP kernel shall contain the mechanisms for implementing the elements of the system and the security policy for controlling access to these elements. Any policy that influences the allocation of physical resources need not be in the SFEP kernel. Unless performance requirements demand it, the policy functions shall not be in the kernel. However, since the actual allocation of resources must be performed by the kernel in a secure and correct manner, it is necessary to have external kernel functions that communicate policy decisions made outside the kernel to the implementation mechanisms within the kernel.

The general security versus policy guidelines shall be followed in SFEP software design. As an example of an application of this philosophy, consider process scheduling. The SFEP software design shall make a distinction between the scheduler (i.e., code that implements the policy that selects the next process to run) and the process multiplexor (i.e., code that implements the mechanism that binds a process to hardware). The correctness of the scheduler is

3.7.2.1.3 Security Versus Policy (Continued)

not necessary for security, thus the scheduler would not be part of the security kernel but would reside in the uncertified operating system.

3.7.2.2 Kernel

The security kernel is the software portion of the SFEP reference monitor. The kernel, supported by the SPM, shall enforce the necessary authorized access relationships between subjects and objects in order to maintain a secure environment. Kernel requirements are specified in the following subparagraphs.

3.7.2.2.1 Process Control

3.7.2.2.1.1 Process Definition

A process shall be identified by a system-wide unique identifier and consists of:

- an execution point defining the virtual memory location of the currently executing machine instruction and the effective ring of execution.
- a known segment table (kst) defining all segments the process is currently "using"
- a known device table (kdt) defining all devices the process is currently "using"
- a message queue holding messages sent to this process by itself, other processes, or devices
- a trap vector defining execution points and trap information save areas for well-defined traps which occur synchronously in process virtual time

3.7.2.2.1.1 Process Definition (Continued)

- a virtual time clock
- a virtual timer by which the process may keep watch on its own use of the processor
- a real timer by which the process may time-out on its operation or the operation of devices it is using

A process has two attributes that are visible to other untrusted processes (subject to security and integrity rules): its security level and its integrity level. These are assigned to to it permanently when it is created, and never change.

Processes are aware of each other by process unique ID, and communicate with each other by sending messages (uninterpreted by the kernel) to each other in typical block/wakeup fashion, subject to security and integrity level restrictions enforced by the kernel.

Each process has a private workspace (or virtual memory and virtual device space) of segments and devices it is referencing frequently. Processes reference segments or devices by their system-wide unique IDs to obtain their attributes or to add them to their local workspaces. Once a process has added a segment or device to its workspace, the process can then reference it by workspace index.

3.7.2.2.1.2 Process Multiplexor

The Process Multiplexor shall be responsible for multiplexing processors among the coexisting processes. Processes coexist in one of three execution states: running, ready or blocked. A running process is one that is currently executing on a processor; a ready process is one that would be running if a processor were available for it to run on; and, a blocked process is one that cannot make immediate use of a processor (even if one were available) because it is waiting for an event to occur - upon occurrence of the event, a blocked process becomes ready.

The Process Multiplexor's tasks shall center around the maintenance of a list of the coexisting processes called the Active Process Table (APT). For each process on the list, the Process Multiplexor associates the current execution state and other necessary data in order to support process dispatching. For example, for each blocked process, the APT contains an identifier for the event being waited for. When the event occurs, the interrupt handler shall notify the Process Multiplexor which may then alter the code for the execution state of the appropriate process from waiting to ready. The scheduler may then make a policy decision based on process priority as to which process should be running; the designated process is then dispatched by the Process Multiplexor.

3.7.2.2.1.2 Process Multiplexor (Continued)

In addition to process dispatching, the Process Multiplexor shall provide the mechanisms for creating/deleting processes and synchronizing cooperating processes as specified in the following subparagraphs.

3.7.2.2.1.2.1 Create/Delete Process

The Create/Delete Process mechanism shall only be utilized by the SFEP Answering Service process to create a process per device structure; i.e., creating a virtual machine environment per interactive user session with the central system. Upon successful completion of the login sequence (validated by the central system), the Answering Service shall spawn a user process via the Create Process function. Upon notification of process completion (as a result of a user logout or hang-up), the Answering Service shall destroy the user process via the Delete Process function.

The Create Process function shall generate the appropriate descriptor tree (including DBR) to provide a spawned process with a unique identity and allow it to be dispatched by the Process Multiplexor. The descriptor tree shall describe the resources required by the process to perform its assigned task (e.g., message handling).

The Delete Process function shall delete the descriptor tree (including DBR) for the appropriate spawned inferior process. Resources described shall be made available to subsequent processes to be spawned (e.g., new users entering the system).

3.7.2.2.1.2.2 Dispatch

The Dispatch function shall be responsible for initiating execution of a new process; i.e., changing its state from ready to running.

3.7.2.2.1.2.2 Dispatch (Continued)

The dispatch shall be accomplished by transferring the root of the process descriptor tree to the SPM. The dispatch function shall only be utilized by the scheduler to implement its policy decision as to which process is to be activated.

During dispatch, the state of the previous process (i.e., registers) shall be saved and the state of the new process restored.

3.7.2.2.1.2.3 Process Synchronization

The Process Multiplexor shall provide the facilities to allow the sequential processes that coexist in the physical computer system to cooperate. The synchronization mechanism shall be implemented via the Block/Wakeup functions specified in the following subparagraphs.

3.7.2.2.1.2.3.1 Wake-Up

The kernel shall provide two wake-up mechanisms in support of process cooperation: an explicit wake-up and an implicit wakeup. The explicit wake-up function shall allow one process to send a message to another by calling the kernel directly. The kernel shall insert the message into the receiver's message queue upon verifying that the security level of the sender is less than or equal to the security level of the receiver.

The implicit wake-up function, Device Wake-up, shall be provided by the kernel to support interrupt handling as specified in 3.1.1.2.2.3. Upon receipt of a device interrupt, the kernel shall associate the interrupting device with a process and insert a interrupt notification message in its message queue.

3.7.2.2.1.2.3.1 Wake-Up (Continued)

The Wake-up functions shall be responsible for insuring that processes waiting for events shall have their states appropriately changed from waiting to ready in order to have them eligible for dispatch.

3.7.2.2.1.2.3.2 Block

The kernel shall provide two block mechanisms in support of process cooperation. Both block functions shall allow a process to read its message queue; if a message exists, the caller is provided with the message by the kernel. However, if no message exists, then one block function shall result in the process losing the processor (BLOCK function). The BLOCK function shall be responsible for changing the current process state from ready to waiting and notifying the scheduler to select the next running process. Another block function (INTERROGATE function) shall merely notify the caller that its message queue is empty; no implied willingness to lose the processor is to be assumed.

3.7.2.2.1.2.4 Trap Handler

As specified in 3.1.1.2.2.3, the security kernel shall be responsible for basic trap handling. The kernel shall be solely responsible for specifying all trap vectors and storage area pointers for each process. However, the kernel shall provide the facilities to allow certain types of traps to be handled in outer rings; the functions required to support their functionality are specified in the following subparagraphs. As expected, however, the kernel shall process all security-related trap conditions (i.e., SPM traps) within ring-0 and shall insure that no trap condition be handled in a ring of less privilege than that possessed by the execution point of the process at the time of trap.

3.7.2.2.1.2.4.1 Set Trap Handler

The Set Trap Handler function shall allow the caller to specify an outer ring trap handler and associated trap save area for a designated trap type. The kernel shall insure that the level of privilege of the user supplied trap handler is greater than or equal to the level of privilege of the requestor.

3.7.2.2.1.2.4.2 Trap Return

To support outer ring trap handling, the kernel shall provide a Trap Return function to allow the kernel to perform required process state modification (necessary to correct the trap condition) and issue the privileged RTT instruction on the user's behalf. The kernel shall insure that the level of privilege specified by the return is less than or equal to the level of privilege of the outer ring trap handler.

3.7.2.2.1.2.5 Clock Management

Since denial of service is not a security issue and the system clock is an extremely noisy information channel, clock/timer functions should be centralized within the Clock Manager component of the operating system without requiring supporting kernel functionality. However, the Level 6 architecture forces kernel clock management support functions as specified below. The kernel must manage both the system clock and watchdog timer since both reside physically within the same segment as trap and interrupt vectors (i.e, segment 0) which are required to be in kernel space only. Thus, the clock/timer must also reside strictly within kernel space since the segment is the basic unit of protection. CPU-generated addresses relating to both the timer and the clock shall be

virtual, but shall always be mapped into the same physical addresses regardless of which process is currently running. The following clock/timer functions shall be supported by the SFEP security kernel:

. Read Real Clock - Allows the system-wide real time clock to be read

. Set Real Timer - Allows a real timer to be created to provide a wake-up to process in specified time period.

. Read Virtual Clock - Allows the process local virtual time clock to be read.

. Set Virtual Timer - Allows a virtual timer to be created to provide a trap to process in specified virtual time period.

3.7.2.2.2 Segment Control

A segment is identified by a system-wide unique ID and is a fixed-length vector of words of memory. (A "word" is the smallest separately protectable unit of virtual memory.)

A segment has several attributes that are visible outside of the kernel (subject to security and integrity rules): its length, its security and integrity levels, and its ring brackets, which define the effective execution rings of processes and devices necessary to perform read, write, execute or call operations on the segment's contents. These attributes are set at the time a segment is created, and cannot be changed.

A process may obtain the attributes of a segment, delete a segment, or add a segment to its kst (initiate it) by supplying the segment's unique ID. Once initiated, a process may reference the segment's contents by index in its kst.

3.7.2.2.2 Segment Control (Continued)

A process may create a segment by specifying its intended length, security and integrity levels, and ring brackets. If the segment can be created, the kernel sets its contents to a predefined initial value (zero) and returns its unique ID.

When a process initiates a segment, it specifies the desired kst index it wishes for the segment. This index must not already be used for another segment. The process' allowable access permission (read, write, execute) on a segment's contents is computed when the segment is initiated. A process may request less access permission than that allowed by security and integrity rules when it initiates a segment.

When a process terminates (opposite of initiate) a segment, the kst index for that segment becomes available to be used for another segment.

When a process deletes a segment, the segment is also terminated in the kst's of all processes having this segment initiated.

The segment control function to be provided by the security kernel are specified in the following subparagraphs.

3.7.2.2.2.1 Create/Delete Segment

The Create Segment function shall create a segment of a given length, security/integrity level, and ring brackets. The kernel shall assign the segment a unique global identifier (returned to caller) and insert the segment name and its attributes into the system global segment catalog within the kernel data base. The privilege level of the created segment shall not be less than the privilege level of the creator.

The delete segment function shall delete the segment with the given global identifier from all initiator process' local space and shall delete the segment from the system global catalog. The privilege level of the deleter shall not be less than the privilege level of the segment.

3.7.2.2.2.2 Initiate/Terminate Segment

The Initiate Segment function shall make a segment known to the requesting process, i.e., move the segment into the process virtual space. The security kernel shall insure that the requested access permission to the segment is consistent with both the privilege level of the requesting process and the privilege level of the segment as defined in the global segment catalog.

The Terminate Segment function shall remove a segment from the process virtual space. There are no contraints to be imposed by the security kernel other than the local segment identifier be valid and the segment not be wired.

3.7.2.2.2.3 Get Segment Attributes

The Get Attributes function shall provide the visible attributes of a segment to a requestor whose privilege level exceeds that of the segment.

3.7.2.2.2.4 Wire/Unwire Segment

The Wire Segment function shall prevent a process local segment from being moved within its virtual space; i.e., the segment is fixed to physical memory.

The Unwire Segment function shall release a previously wired segment allowing its movement within process virtual space.

3.7.2.2.2.5 Primary Memory Manager Interface

The Memory Manager portion of the SFEP operating system shall be responsible for all memory control policy decisions. The Primary Memory Manager function within the kernel shall be responsible for carrying out its policy decisions in a secure manner. The kernel memory manager shall insure that some acceptable number of free primary memory blocks always exist. In general, whenever the number of free primary memory blocks drops below the acceptable limit, the operating system shall be awakened to determine which pages should be transferred to mass store. However, within the SFEP environment, no supporting mass store is required. Thus, this general functionality need not be implemented, yet the primary memory manager function within the kernel should not prevent its implementation.

3.7.2.2.3 Device Control

A device is identified by a system-wide unique ID. From the viewpoint of a process accessing a device, a device consists simply of a single "contents word", which the process may read or write. All finer-grained indexing of various parts of a device (status and control registers, storage media contents, etc.) is accomplished by device-specific protocol of reading and writing this contents word.

A device has several attributes that are visible outside the kernel (subject to security and integrity rules): its security and integrity levels, and its ring brackets, which define the effective execution rings of processes necessary to perform read, write, and control operations on this device's contents. These attributes are set by the kernel.

A process may obtain the attributes of a device, or add a device to its kdt (initiate it) by supplying the device's unique ID. Once initiated, a process may reference the device's contents by index in its kdt.

3.7.2.2.3 <u>Device Control</u> (Continued)

When a process initiates a device, it specifies the desired kdt index it wishes for the device. This index must not already be used for another device.

When a process terminates (opposite of initiate) a device, the kdt index for that device becomes available to be used for another device.

From a security and integrity viewpoint, all I/O device operations are always both read and write. This is true because all operations may return status information visible to a using process, and may result in a change of state visible to the external world (e.g., line turn-around). Therefore, a process must have security and integrity levels equal to those of a device to initiate it. The kernel assigns and changes the security and integrity levels of devices as necessary to allow them to be initiated by processes that need to use them.

A device may only be initiated in one process' kdt at a time. Any simultaneous sharing of devices among processes is accomplished outside the kernel via interprocess communication.

The kernel supports a form of "write-ring" protection on the actual data (as opposed to status and control information) handled by the device. All the operations that a process may perform on a device are partitioned into four classes: trusted control, normal control, write and read. trusted control operations are those operations on a device

3.7.2.2.3 <u>Device Control</u> (Continued)

that can potentially invalidate the security and integrity checks made by the kernel (e.g., changes to device controller microcode), and therefore, must be restricted to trusted code. Normal control operations are those that need to be available whether or not the process is restricted to only read or only write the data handled by the device (e.g., positioning operations, echoing, and status returns). Write operations include actual writing of the data of a device, and control operations that should not be available to a process restricted to reading (e.g., erasing a tape). Read operations include actual reading of the data of a device.

For each device type catalogued in the system, the kernel maintains a map of the I/O operations in each of these classes. When a process initiates a device, it may request read-only, write-only, or read-write permission. After initiation, the process may perform all normal control operations, and either or both of the read and write operations defined for the device.

There are also two generic types of I/O operations a process may perform on an initiated device: synchronous and asynchronous. Synchronous operations are performed directly by a process, and take effect and return values immediately. All control operations and some data operations are synchronous. Asynchronous operations are started by a synchronous set-up operation, and cause a device to operate on behalf of, using the virtual memory

3.7.2.2.3 Device Control (Continued)

space of, and asynchronously with, the starting process. Normal data read and write operations are asynchronous. The effective ring used by a device accessing a segment during an asynchronous operation is that of the process when it started the operation. Only one asynchronous operation may be in progress on a device at a time. Some synchronous operations (e.g., status) may be performed on a device with an asynchronous operation in progress.

A device communicates with its using process (to indicate termination of an asynchronous operation, or to signal events that require the process' attention) by sending the process its device unique ID as a message. The process can then perform operations on the device to obtain more detailed information.

The device control functions to be provided by the security kernel are specified in the following subparagraphs.

3.7.2.2.3.1 Add/Remove Device

The Add/Remove Device functions shall only be used by a trusted configuration process to add or remove physical devices (and their associated attributes) from the kernel data base.

3.7.2.2.3.2 Initiate/Terminate Device

The Initiate Device function shall cause an I/O device (or channel) of the class specified to be allocated to the invoking process. All devices shall be allocated at an access level equal to that of the requesting process. Each process shall have an associated list of currently

3.7.2.2.3.2 Initiate/Terminate Device (Continued)

"active" I/O devices which it has requested and received access to.

The Initiate Device function shall verify that the requesting process is eligible to receive the requested device (channel) and that the device (channel) is not currently assigned to another process. The kernel shall insure that only one I/O descriptor per device (channel) exists.

The Terminate Device function shall return the specified device to the device available pool. The I/O descriptor for the specified device (channel) residing in the requesting processes I/O descriptor tree shall be deleted.

3.7.2.2.3.3 Send/Receive Message

The Send/Receive Message functions shall provide the SFEP kernel to central system kernel communications channel. Essentially, these functions shall provide the Interface Unit (IU) driver and associated resource manager.

Messages of up to one segment in length shall be transferable between SFEP and the central system. This interface shall support "scatter-gather" list processing, while allowing IU operation as either a mapped or premapped device. The list processing is required strictly to support the central system. The scatter-gathering on the SFEP side shall be provided by the SFEP I/O virtual mapping mechanism, i.e., all SFEP accesses are logically contiguous, but physically scattered. The central system has no corresponding mapping mechanism; thus, absolute physical addresses are required which

3.7.2.2.3.3 Send/Receive Message (Continued)

necessitates list processing to minimize interrupt handling overhead.

To minimize SPM functionality, the IU shall support two distinct channels: set-up and connect. The setup channel shall be used to notify the IU of the location (virtual) of the control list, while the connect channel shall be used to notify the IU of the location (virtual) of the SFEP data area. Thus, the IU may access both the control list and data area in a mapped manner.

The IU control list shall reside exclusively within kernel space, while the message buffer shall reside in user space. No copying of a message into kernel space shall be required in order to initiate a transfer.

These kernel interface functions shall support both binary format and character format transfers. Binary transfers shall be performed only in multiples of four central system words; character transfers shall be performed in multiples of a full central system word.

Management of the interface resource shall be on a first-in/first-out basis.

3.7.2.2.4 Bootload

SFEP shall be bootstrapped from the host processor through the 6000 Series 60 Interface Unit. During the initial stage of bootload, a portion of the security kernel shall be transferred into SFEP memory. The initial portion of the kernel shall be sufficient to initiate any remaining

3.7.2.2.4 Bootload (Continued)

memory load required through the IU. In Figure 14 is shown the contents of memory following the initial memory load. In Figure **15**, a DBR, two I/O descriptor, two memory descriptors and a procedure segment have been loaded. The DBR establishes the trees of I/O and memory descriptors. The first I/O descriptor establishes the SPM as a device; the second establishes the IU for further memory loading. The first memory descriptor establishes the loaded procedure; the second establishes a memory area for further input. The processor Program Counter shall be set to extract the first order of the procedure segment. The SPM shall load the DBR from a predefined memory address as directed by the bootload command line. The current ring shall be initialized to the kernel domain, ring-0. The contents of the program counter is a virtual address and the corresponding address shall be fetched from memory using the initial DBR and memory descriptors. Processing shall continue in ring-0 with all addresses interpreted as virtual addresses until explicitly changed by software.

In support of stand-alone secure computer applications, the SFEP bootload procedure shall allow replacement of the central system by a standard Level 6 peripheral (i.e., diskette) without bootload redesign. That is, only replacement of the IU driver and its associated descriptor with a diskette driver and diskette descriptor shall be required for conversion.

3.7.2.2.4 Bootload (Continued)

SFEP bootload/initialization may be accomplished in two ways depending on system initialization philosophy. The particular mode to be implemented, as specified in the following subparagraphs, shall be application dependent. SFEP shall utilize the static initialization approach; however, SFEP design shall not preclude implementation of dynamic initialization.

3.7.2.2.4.1 Static Initialization

In the static initialization method of bootload, the process structures required for SFEP operation are supplied; e.g., from a previous system. That is, the bootload modules would manifest a fully initialized system, rather than letting the system bootstrap itself in a complex way each time it is loaded.

3.7.2.2.4.2 Dynamic Initialization

In the dynamic initialization method of bootload, the kernel is responsible for constructing the process structures required for SFEP operation. This action requires the support of appropriate kernel primitives to create the necessary environment.

3.7.2.3 Operating System (OS)

Supervisory control functions for SFEP on-line operational software shall be comprised of a combination of executive modules, special processes, and distributed kernel functions. Some normal OS functions will be implemented in the kernel, descriptions of these are included in the OS specification below.

- Process scheduling (OS process)
- Process dispatching (distributed kernel function)
- Interrupt interception and routing (distributed kernel function)
- Trap interception and routing (distributed kernel function)
- Interprocess communication and synchronization (distributed kernel function)
- Real-time clock management (distributed kernel and OS functions)
- Memory management (distributed kernel function and OS process)
- Intra-process task management (distributed OS function)

This functionality shall support a multiprogramming with priority processing environment in which all operational software is resident in primary memory. Multiple processes (isolated and protected by kernel functions and data) shall time share access to common executive and application functions written in reentrant procedure code.

3.7.2.3.1 Process States

Since several processes must time share system time and resources, each process may exist at different times in different states or stages of processing.

A process may be in one and only one of the following logical states:

- Dormant - The process does not exist in the kernel process catalog. The process may physically exist in the system, but it may not be scheduled for execution until it has been "created" by the kernel as requested by another process.
- Ready - The process exists in the process catalog, is scheduled for execution and is not waiting for any event to be completed other than the currently executing process to block itself or a system time-out to preempt it.
- Running - The process has been dispatched by the kernel and is executing.
- Waiting - The process has blocked itself awaiting the completion of some other event. That event may be an input/output operation in progress or the completion of another process execution cycle.

Figure 16 illustrates these four process states and the kernel gates used to effect changes in process state.

3.7.2.3.2 Process Scheduling

Prime responsibility for process control shall reside in the kernel; however, to make provision for varied scheduling policies without impacting certified kernel code, the scheduling function shall be divided into a set of kernel functions and a separate isolated operating system process. The kernel shall maintain all security sensitive system level process state and status information needed by the OS scheduling process to make scheduling decisions according to operations policy. The kernel functions shall maintain lists (queues) of processes by process state as defined above. The process "ready" lists shall be accessible by the OS scheduler process.

The OS scheduler shall perform two basic operations:

1. Select the next process to be run based on ready process priority.
2. Optionally assign a watchdog, time-out to the selected process. A watchdog time-out shall serve two purposes: (1) to force a process block and (2) to execute a wake-up operation.

The scheduling policy for SFEP operations shall require scheduling decisions to be based on a first-in/first-out (FIFO) within priority scheme.

3.7.2.3.3 Process Dispatching

The scheduler process shall be dispatched by the kernel whenever a running process "blocks" itself. The scheduler will then decide which process is to run next and call the kernel to execute the dispatch operation. Figure 17 depicts the basic scheduling mechanism.

3.7.2.3.4 Interrupt Interception and Routing

All interrupts shall be intercepted by the kernel and either handled within the kernel or communicated to the associated process. When an interrupt occurs, the firmware shall direct a jump to the kernel interrupt service function. The kernel shall record the interrupt occurrence in an IPC and wake-up to the associated process and then execute a level change to cause the interrupted process to be reactivated. When the process associated interrupt is activiated, it must test for the occurrence of the interrupt by checking its message queue.

3.7.2.3.5 Trap Interception and Routing

All traps shall be intercepted by the kernel and either handled within the kernel or communicated to the associated process. The method of communication shall be the same as with interrupt handling except that for traps, the kernel shall provide the associated trap handler with a copy of the trap save area.

3.7.2.3.6 Interprocess Communication and Synchronization

Interprocess communication and synchronization shall be provided via the following kernel mechanisms:

- Wake-up - A running process signals the kernel that it wants another process to be activated. The wake-up causes the requested process to be placed on a scheduler ready queue.
- Block - A running process signals the kernel that it is ready to give up the processor to await some action by another process or completion of an I/O operation.
- Interprocess Communication - A message queue is maintained by the kernel for each running, ready or waiting process for the purpose of conveying information from other processes to that process under kernel control.

3.7.2.3.7 Real Time Clock Manager

A Real Time Clock Manager (RTCM) functionality shall be provided by which system and applications functions can obtain real time readings, one-shot time-outs and cyclic time-outs. Additionally, the RTCM shall provide for maintenance of virtual time-outs; that is, time-outs within a given process function only when the process is running.

3.7.2.3.7 Real Time Clock Manager (Continued)

The kernel shall contain that part of the RTCM which intercepts the RTC interrupt and updates the distributed system clock. A distributed OS module shall be responsible for processing time-of-day, real time time-out, and virtual time-out requests.

3.7.2.3.8 Memory Manager

The memory manager shall consist of two parts, one a distributed kernel function that essentially provides a buffer management capability, the other an Operating System process that supports the kernel in maintaining an adequate supply of buffers.

The buffer manager capability shall enable a user process to "get" and "release" buffer space as needed. The kernel shall control the assignment of buffers by maintaining lists of assigned buffers by process and a list of free buffers. When a process is made "ready" by the kernel, that process will receive a quota of buffer space that it may request. As buffers are requested by a process, descriptors are added to the process access space. As buffers are released, corresponding descriptors are removed from the access space.

3.7.2.3.9 Intraprocess Task Management

A Task Manager function shall be provided to support application process operations. The Task Manager, in conjunction with the real time clock manager and buffer manager, shall provide the applications programmer with a basic executive capability designed to take advantage of the unique characteristics of the Level 6 computer.

3.7.2.4 Application Software

Application software for SFEP shall consist of Multics Communications and Remote Communications Network interface functions.

The Multics Communications subsystem functions shall perform the processing required to handle Multics on-line interactive communications with remote terminals. The Multics to SFEP interface will be handled by the kernel. Following functions shall be provided.

- Terminal Answering Service
- Terminal Handler
- Communications Network Interface

3.7.2.4.1 Answering Service

The answering service shall function as a separate trusted process to provide dialup, login, terminal handler process creation, and logout functions.

3.7.2.4.1 Answering Service (Continued)

The dial-up function of the answering service module shall perform all processing required to establish telephone line connection between user terminals and SFEP.

The login function of the answering service shall perform all processing required to accept the user login message and determine its destination. Login messages for Multics shall be directed to Multics for validation.

Dialogue with the user terminal shall be maintained to communicate disposition of the login message. Successful completion of the login sequence shall terminate the login function for that terminal and invoke the terminal handler process creation function.

The terminal handler process creation function of the answering service module shall communicate with the kernel to create a terminal handler process for each successfully logged-in terminal. After successful completion of this function, control shall be passed to the logout function.

3.7.2.4.1 Answering Service (Continued)

The logout function shall monitor the answering service interprocess message queue awaiting a logout message to be transmitted by either the associated terminal handler process or an operating system process. The logout function shall communicate with the kernel to delete the designated terminal handler process.

3.7.2.4.2 Terminal Handler

A reentrant terminal handler module that can be assigned to handle many different terminal sessions concurrently as separate processes (by virtue of isolation guaranteed by the kernel) shall be provided.

3.7.2.4.3 Communications Network Interface

This subsystem shall provide the capability for SFEP accessing a network (such as ARPANET) with message traffic either from the Multics host or from remote user terminals.

3.7.2.5 Support Software

In addition to operating system, kernel, and communications functions required to provide the prime on-line SFEP functionality, a subsystem of support modules are required to provide the following capability:

- Operator console communications
- Configuration support
- Debug support
- Test and diagnostics

3.7.2.5.1 Operator Console Communications

This module shall provide the basic capability for the operator to interact with both on-line and off-line modules to send and receive information via the operator console device. Use of this module shall be restricted to SFEP software communications not related to remote terminal communications other than auditing functions.

3.7.2.5.2 SFEP Initialization Module

This module shall function at system start-up and prior to any remote terminal operations, to initialize SFEP operational software parameter tables and system configuration tables. Multics shall transfer the desired configuration data to SFEP, whereupon the initialization module shall proceed to determine the real attainable configuration. The initialization module shall then, if necessary, interact with the operator through the operator console device to resolve any configuration assignments. The revised configuration data shall be transmitted back to Multics for approval and/or further directed initialization action.

As part of the initialization function, all MLCP communication control programs and line control tables shall be loaded into the MLCP memories.

3.7.2.5.3 Debug Module

The SFEP debug module shall provide for interactive off-line program check-out operations for troubleshooting new programs and making trial program corrections. Debug functions shall enable the programmer to display and modify the contents of memory locations and the arithmetic and base registers. Capability shall also be provided for automatic activation of the debug module through selectable breakpoints placed in the program under test via debug operations. Capability to restart the test program shall be provided.

The debug module shall be capable of functioning under kernel control and may co-exist with a test process within its access space.

3.7.2.5.4 Audit Log

This support module shall provide, during on-line communications operations, the capability to monitor system activity. The following events shall be output to the console printer:

- Terminal login
- Terminal logout
- Out-of-normal conditions
  - Terminal errors
  - Line errors
  - System faults
  - Access violations
- Log of accesses to classified segments
- Change in privilege

3.7.2.5.5 SFEP Memory Dump Module

This support module shall enable the SFEP operator to selectively dump areas of main memory to the console printer in support of system debug operations. Provision shall be made for outputting either hexidecimal or ASCII.

3.7.2.5.6 Test and Diagnostics

SFEP test and diagnostic functions shall provide for off-line SFEP hardware testing as a system confidence test. Diagnostic functions shall provide for testing all SFEP computer programmable hardware, outputting printed or displayed diagnostic results in support of system initialization and maintenance operations.

4.0 QUALITY ASSURANCE PROVISIONS

4.1 General

The Quality Assurance Program to be applied to the SFEP shall be conducted in accordance with the criteria described herein and the SCOMP Product Assurance Program Plan. The SCOMP P.A. Program Plan shall describe the integrated quality and reliability assurance activities applicable to SCOMP prototype and production systems.

4.1.1 Responsibility for Tests

Unless otherwise specified in procurement documentation, the supplier is responsible for the performance of all tests and inspections specified herein.

4.1.2 Special Tests and Examinations

The following requirements of Section 3.0 shall be verified entirely, or in part, by inspection of the equipment and its drawings.

A. (3.2.2) Physical Characteristics

B. (3.3.3) Identification and Marking

C. (3.3.4) Workmanship

D. (3.3.5) Interchangeability and Replaceability

4.1.3 Reliability Analysis

See paragraph 3.2.3.

4.2 Quality Conformance Inspections

4.2.1 Engineering Design Evaluation

4.2.1.1 Hardware Verification

A SFEP logic design verification analysis shall be performed to verify that the hardware portion of the SFEP security requirements are accomplished by the digital logic mechanization of the various modules that comprise an SFEP. The analysis shall consist of two phases. First, development of correspondence between this specification and the detailed specifications using hardware flow charts and a corresponding set of operating specifications which describe elements of the hardware in a simple way. The second phase of the analysis shall consist of detail logic analysis using register and/or instruction level simulation as well as manual analysis.

4.2.1.2 Design Evaluation Testing

4.2.1.2.1 Prototype Development Tests

A prototype SFEP shall be subjected to design evaluation test sequences to verify its functionality and operation under worst case conditions of power,temperature and clock frequency operation. The tests shall be conducted in a minicomputer configuration whose standard functional elements (bus, memory, etc.) have been previously acceptance tested. Security functionality as well as the 6000/Series 60 Interface Unit functionality shall be verified using operating software developed as specified in paragraph 4.2.1.2.2.

4.2.1.2.2 Prototype Test Software

The prototype SFEP shall be development tested using

4.2.1.2.2 Prototype Test Software (Continued)

evaluation software developed with the aid of a simulator. Software developed on this simulator shall test the security functions to insure that the security requirements for the hardware described in paragraph 3.0 are exercised. The test and evaluation software shall also exercise the functionality of the 6000/Series 60 Interface Unit.

4.2.1.3 SFEP Qualification Tests

Structural and thermal environmental qualification tests for the SFEP are not required. Qualification for the SFEP shall be established by structural similarity to ruggedized minicomputer circuit elements upon which tests shall be performed. The similarity units shall include at least one CPU and one 32K word memory.

4.2.2 Prototype Inspection and Test

SFEP prototype subassemblies shall be visually inspected for workmanship, damage and assembly configuration prior to first powered operation.

Prototype SFEP's shall be acceptance tested in accordance with paragraph 4.2.1.2.1.

4.2.3 Production Acceptance Tests and Inspections

4.2.3.1 Inspection Criteria for Aero Fabricated Assemblies

4.2.3.1.1 Workmanship

Workmanship shall be verified on each production assembly to Honeywell workmanship standard, OED 23036 to meet the requirements of MIL-STD-454 Requirement 9.

4.2.3.1.2 Configuration

Each production assembly shall be visually examined in individual parts kit form prior to issuance to assembly and again upon completion prior to acceptance testing. Configuration examination shall include:

- Verification that correct part types have been issued for manufacture.
- Completed assemblies are complete and visually identical to a standard reference assembly or photograph thereof.

4.2.3.1.3 Electrical Parts Inspection

The logic functionality, lack of damage, and marking of integrated circuits to be assembled into production assemblies shall be verified by inspection and test prior to assembly. Appropriate quality control sampling plans based lot total percent defective (LTPD) acceptance criteria shall be employed for marking and damage.

4.2.3.2 Production Acceptance Testing

4.2.3.2.1 Acceptance Tests

Production acceptance tests shall be conducted under the supervision of quality control using approved test procedures, equipment and software. Each assembly shall be accepted with the SFEP unit for which it is intended. Spare assemblies may be acceptance tested in any SFEP if compatible configuration provided that all functional elements used in the test have been inspected in accordance with paragraph 4.2.3.1.

4.2.3.2. Production Test Software
2

Software used for acceptance testing of production SFEP's shall be derived from the prototype software (see paragraph 4.2.1.2.2) or other suitable source which insures that each SFEP function is exercised.

Production test software shall be formally issued and controlled by quality assurance in accordance with Honeywell Design Procedure 3.3

4.2.4 Kernel Verification

The security kernel's verifiability property requires that its correctness be provable in a rigorous manner using a mathematical model as the basis for the criteria to be met. Verification of the SFEP security kernel (i.e., kernel software and SPM) is sufficient in order to verify the security properties of the system since all protection mechanisms are collected within the kernel. The methodology to be followed in verification consists of performing correspondence proofs between levels of hierarchically ordered models of the security kernel (i.e., the Stanford Research Institute technique based on levels of abstraction). The steps to be followed in verification are specified in the following subparagraphs.

4.2.4.1 Mathematical Model

The mathematical model (Secure Computer System: Unified Exposition and Multics Interpretation) rigorously defines the concepts of security and security compromise consistent with the national security classification scheme. The model is a finite state machine model

4.2.4.1 Mathematical Model (Continued)

and defines a set of rules of operation for making state transitions. If the system is initialized to a secure state, then the rules of operation guarantee that all subsequent states are secure. In particular, the basic elements of the model are subjects and objects. Subjects are active system entities such as users or processes which can access system resources, and objects are passive system entities such as program segments and peripheral devices that can be accessed by subjects. The model defines the types of access that a subject may have to an object.

4.2.4.2 Formal Specification

The first step in verification involves the transition from the mathematical model of a secure system to a formal specification of the kernel. The formal kernel specification shall be hierarchical in form. The kernel shall be decomposed into levels of abstract machines each specified as a Parnas module. A correspondence proof between the formal kernel specification and the mathematical model shall be performed.

4.2.4.3 Algorithmic Representation

The second step in verification involves the transition from the formal specification to an algorithmic representation of the security kernel. The algorithmic representation shall specify system functionality by providing required functions, inputs and outputs. A correspondence proof between the algorithmic representation and the formal specification shall be performed. This step, like the preceding one, is a further limitation and definition of the security abstraction.

4.2.4.4 Machine Language Representation

The final major step in verification shall provide proof of correctness for the machine language version of the security kernel. A correspondence proof between the machine language representation and the algorithmic representation shall be performed.

If a compiler is used in this step, it is not necessary to verify the compiler. The compiler's effect on the kernel can be verified by comparing the source code model for each kernel module with the compiler-produced object code implementation.

The final verification step (resulting in a proven machine language representation of the security kernel) assumes specific hardware behavior during execution of the code by the SFEP processor. Thus, a hardware verification phase is also required to satisfy total system verification requirements. Hardware verification requirements are addressed in 4.2.1.1.

5.0 PREPARATION FOR DELIVERY

See 3.2.6.

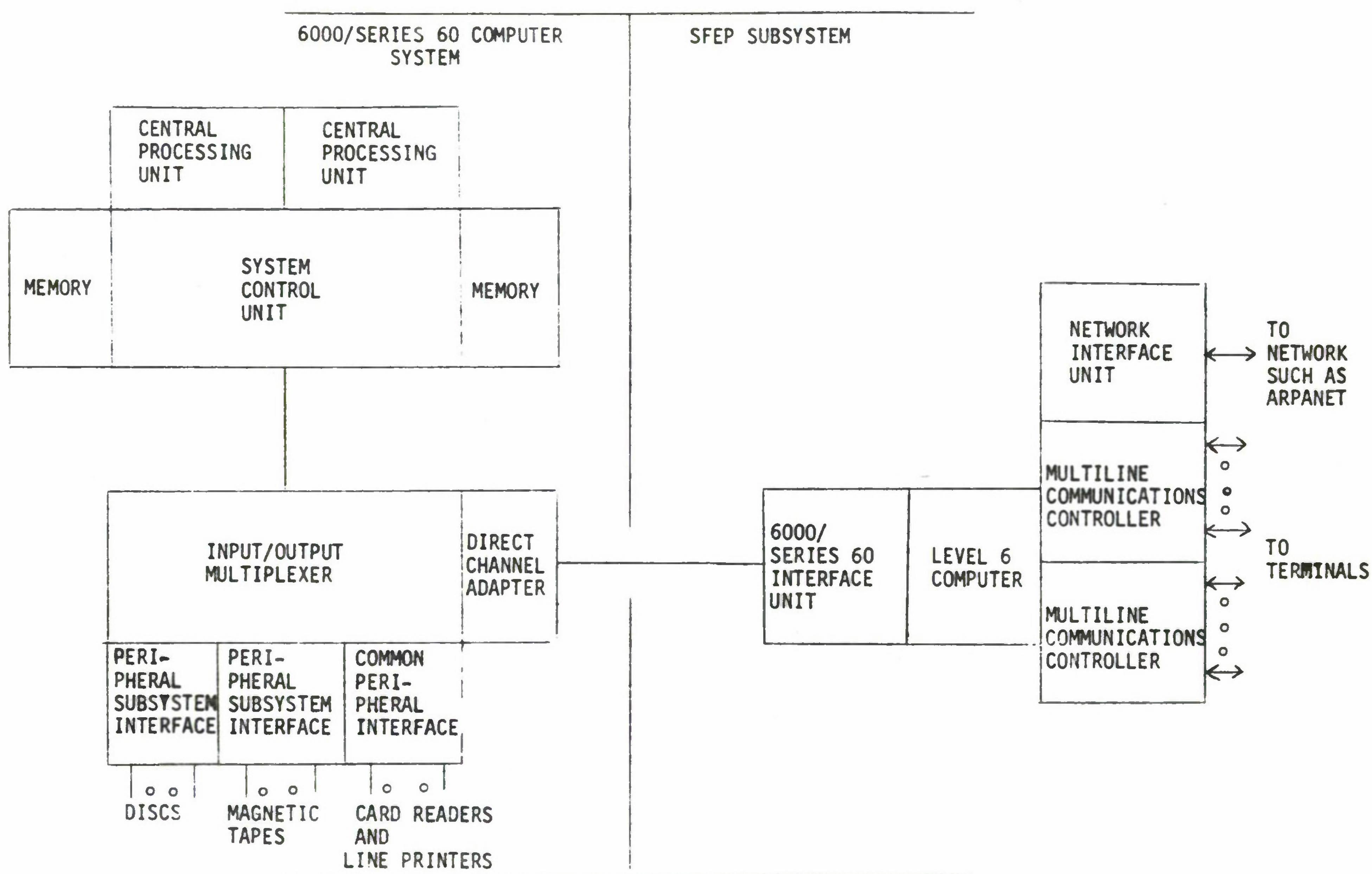


FIGURE 1. <u>6000/SERIES 60 SECURE SYSTEM</u>

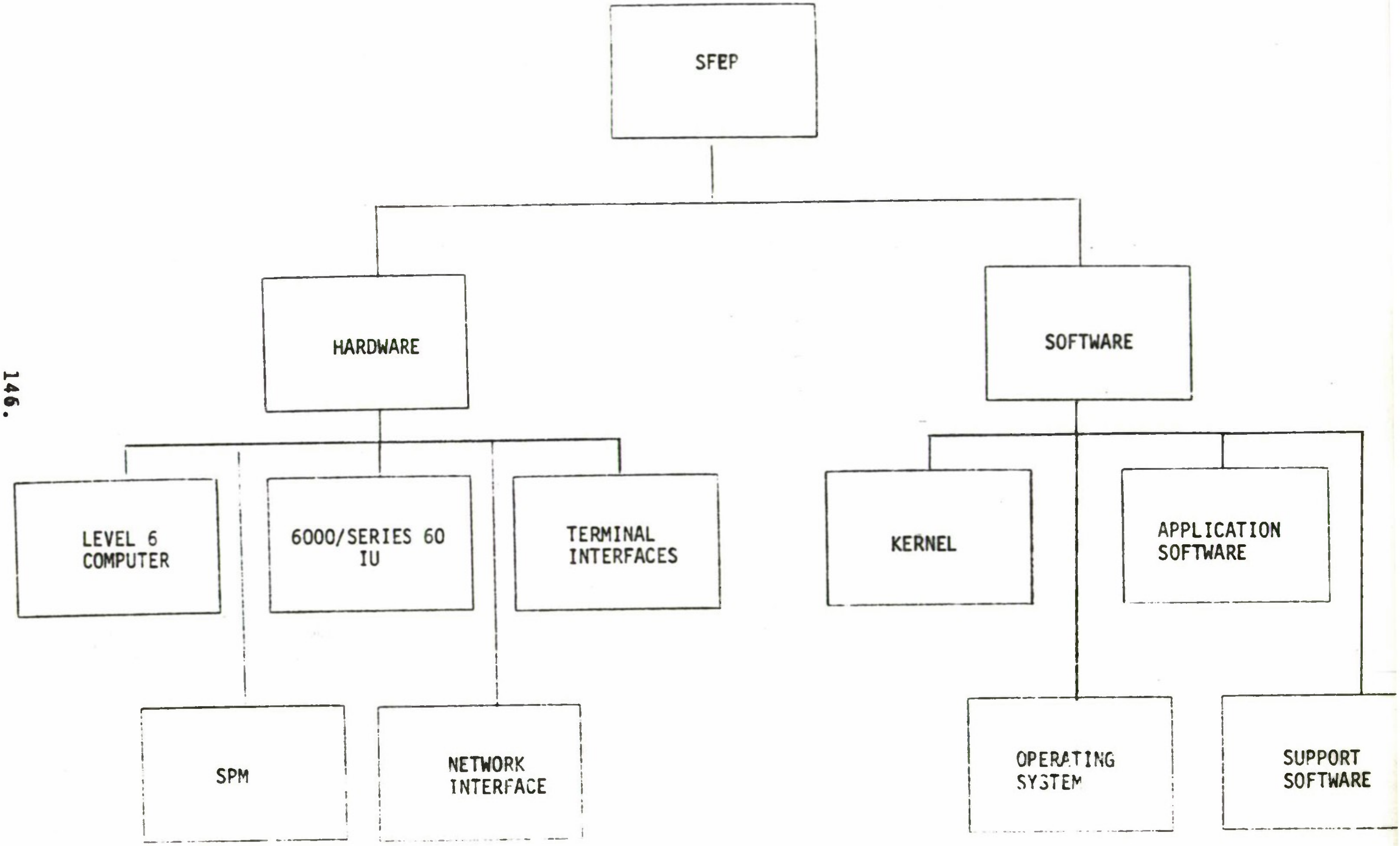


FIGURE 2. <u>SFEP SUBSYSTEM ELEMENTS</u>

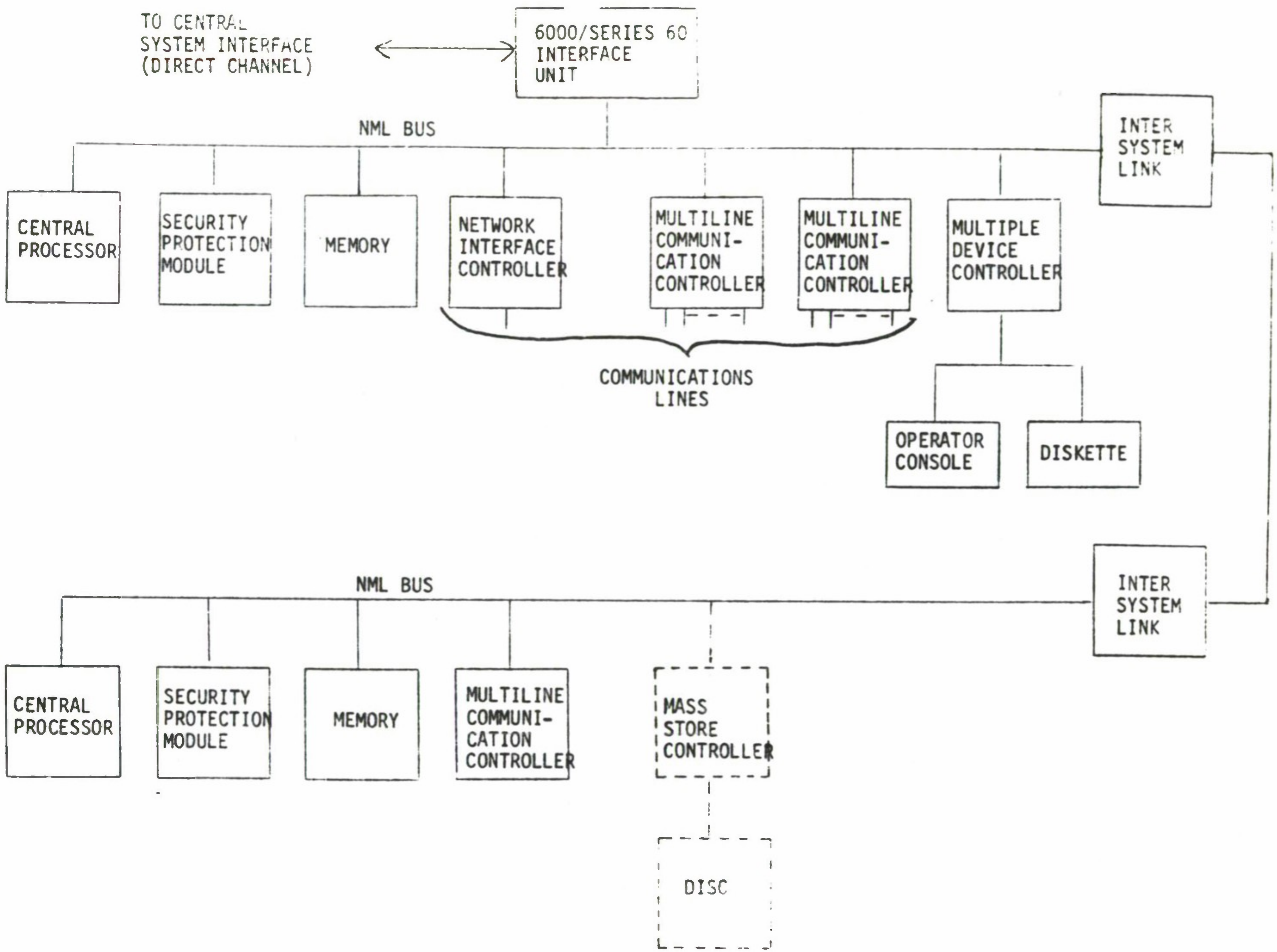


FIGURE 3. SFEP SUBSYSTEM BLOCK DIAGRAM

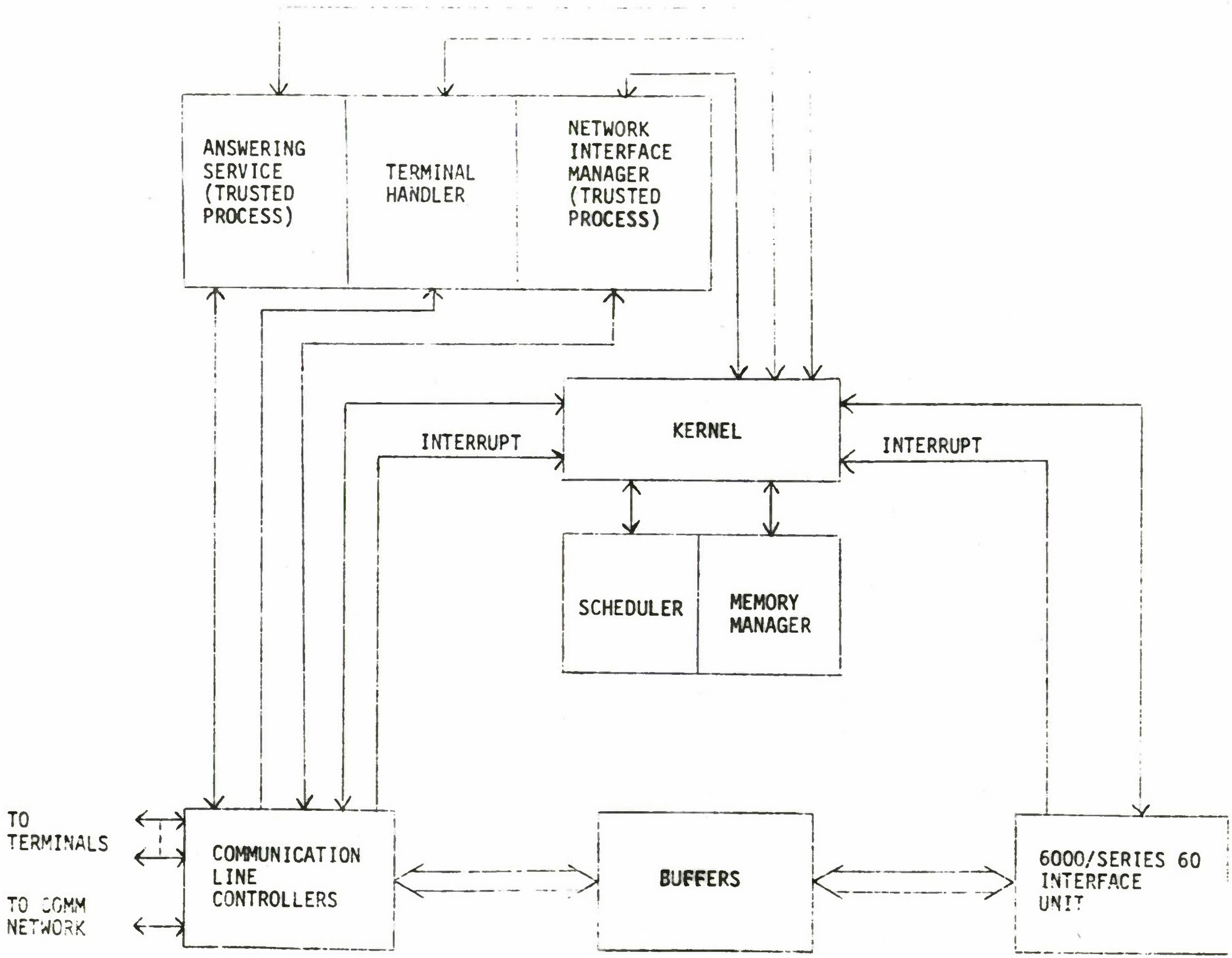


FIGURE 4. SFEP SOFTWARE FUNCTIONAL BLOCK DIAGRAM

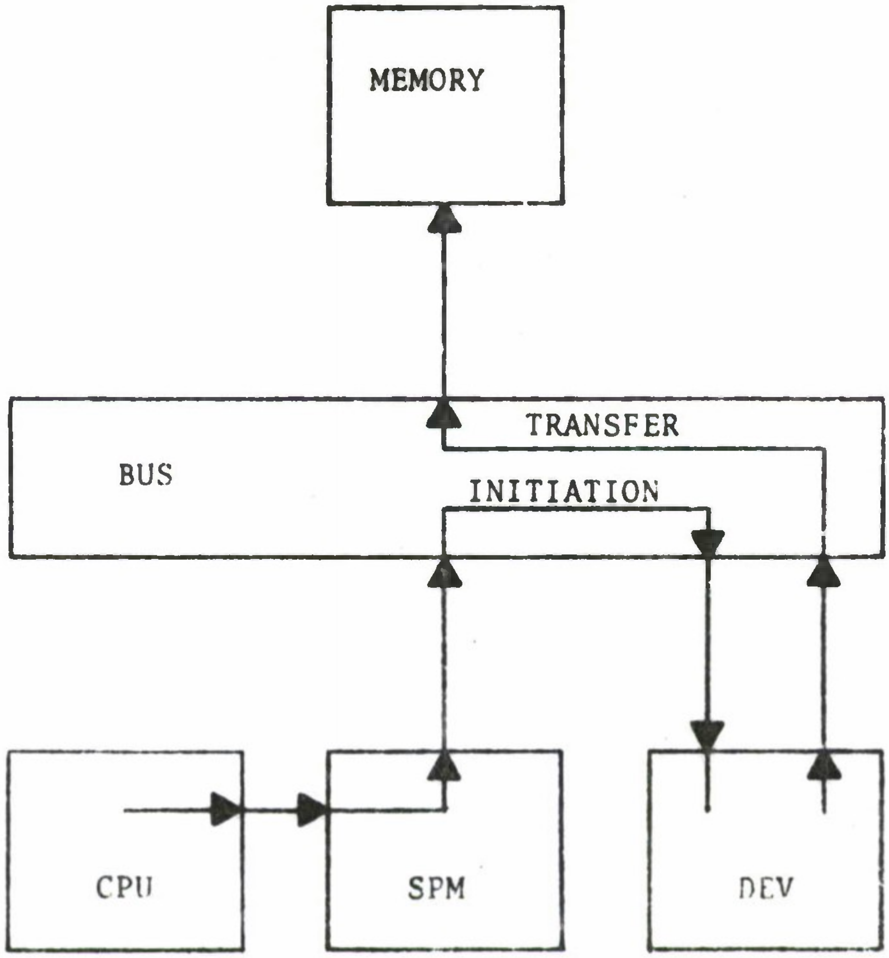


FIGURE 5. PREMAPPED I/O FLOW

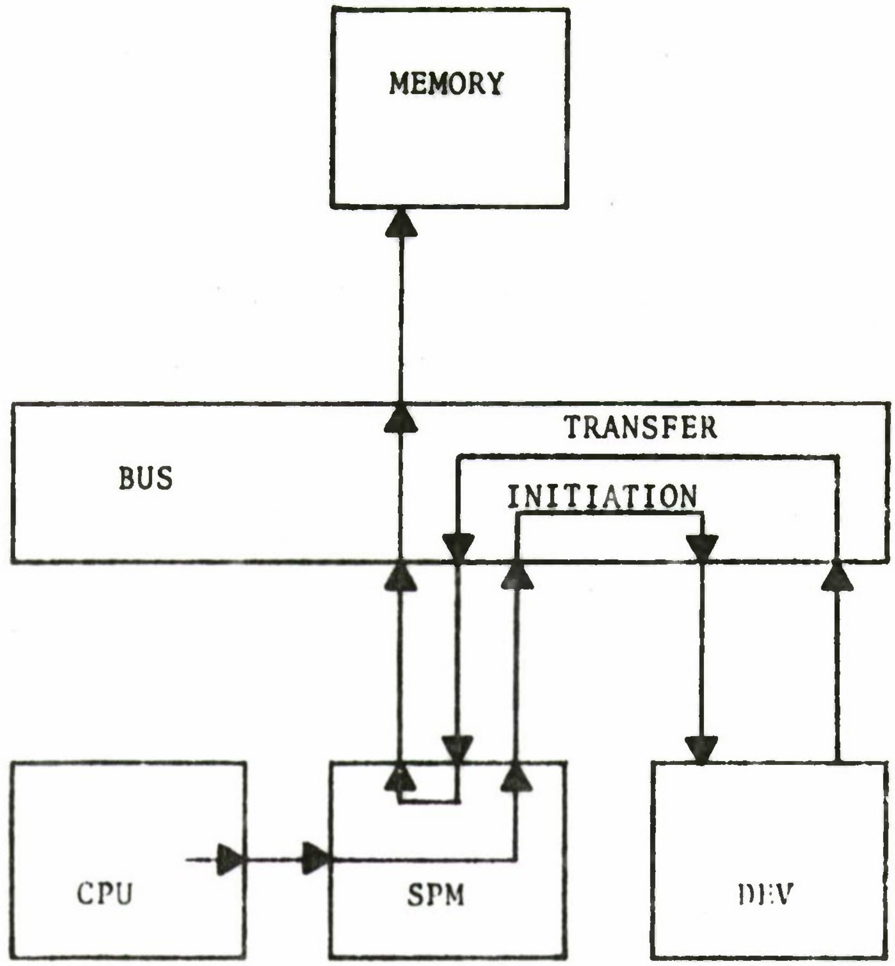


FIGURE 6. <u>MAPPED I/O FLOW</u>

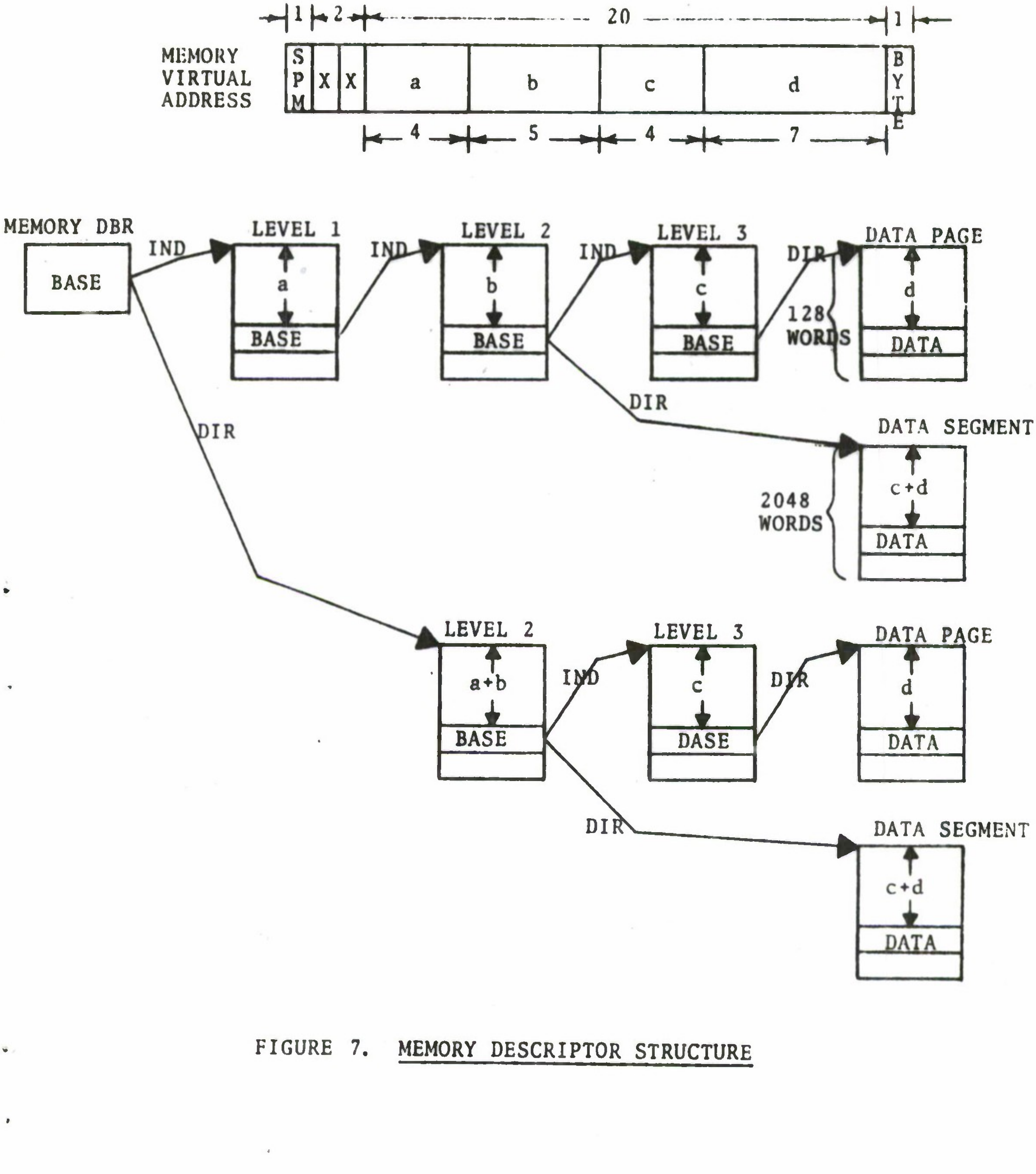


FIGURE 7. MEMORY DESCRIPTOR STRUCTURE

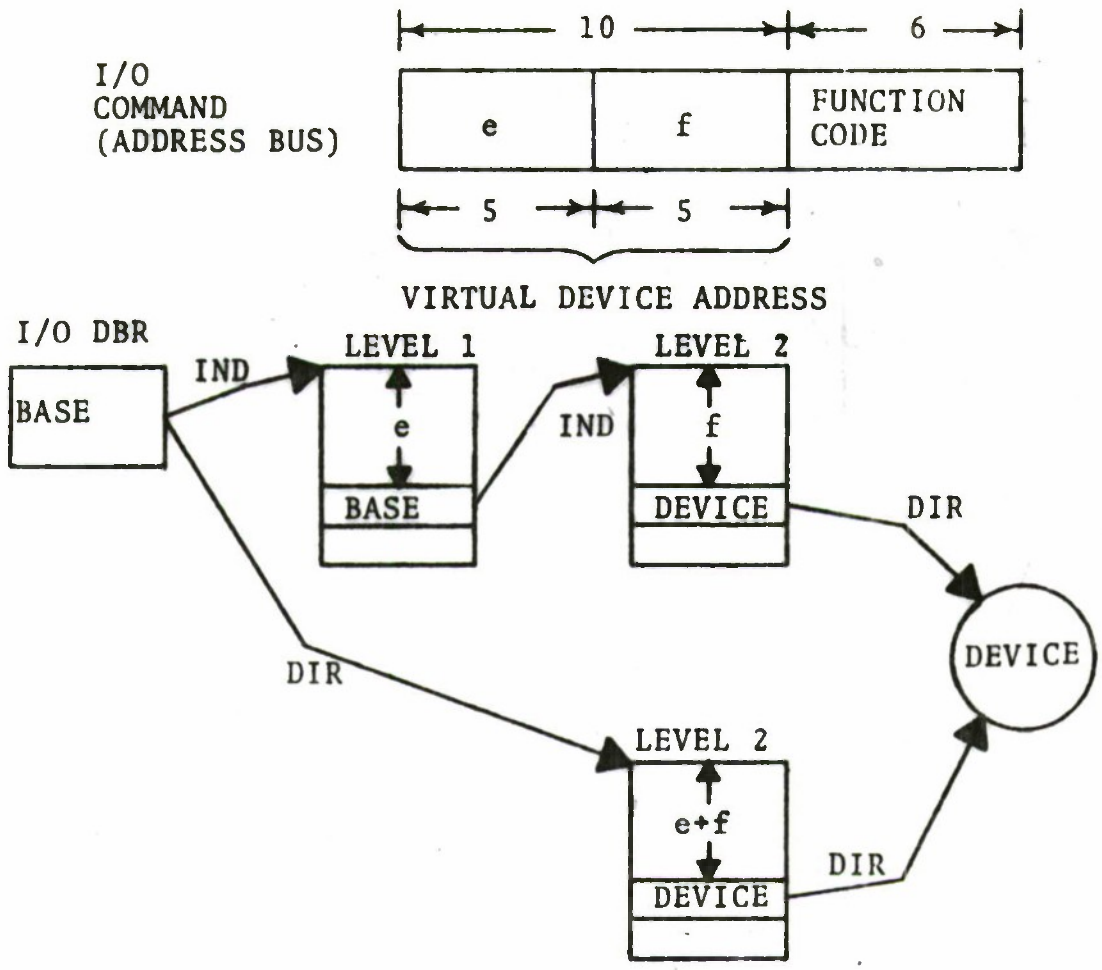


FIGURE 8. I/O DESCRIPTOR STRUCTURE

INTERPRET DRAWING IN ACCORDANCE WITH MIL-D-1000, FORM

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

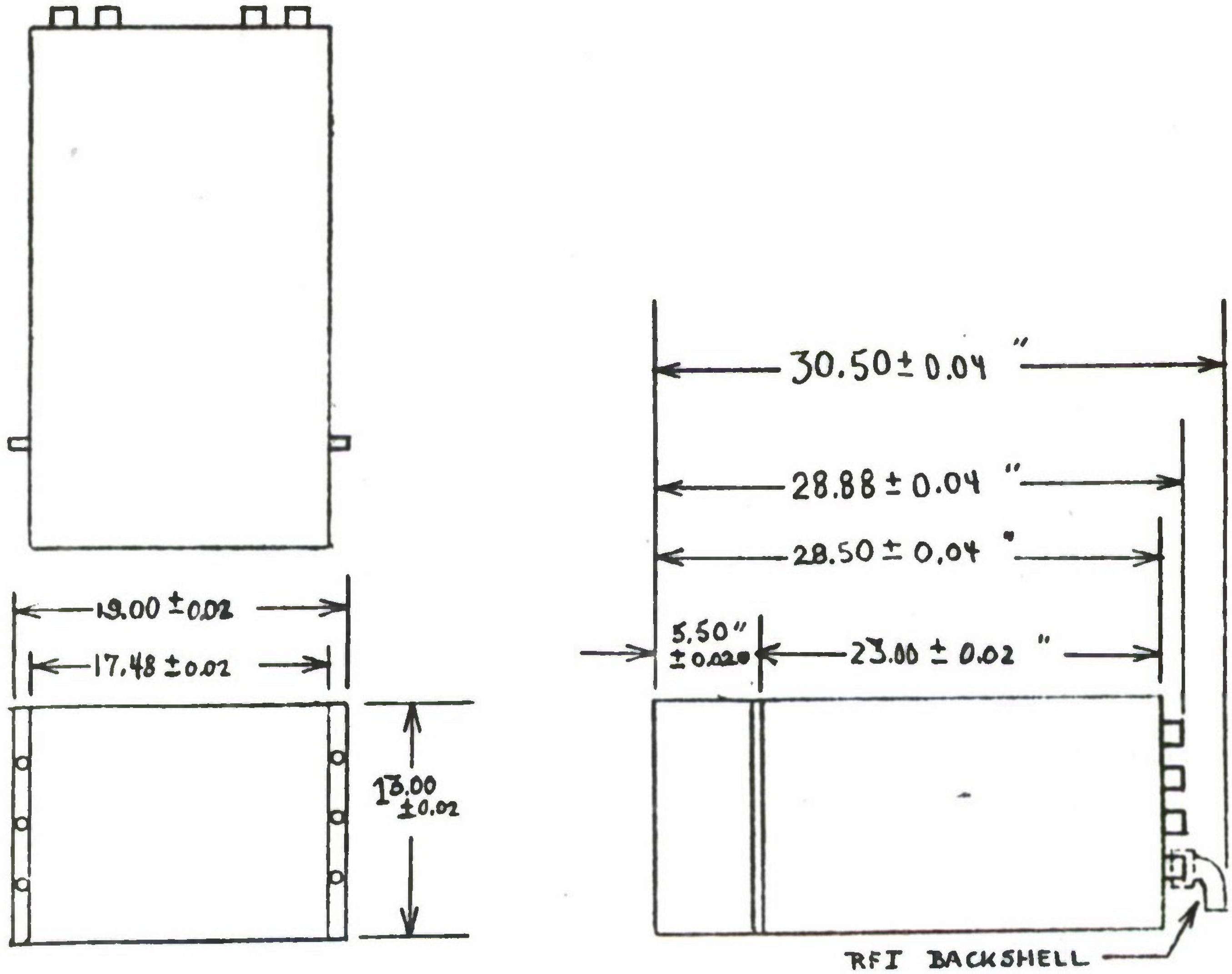


FIGURE 9. SCOMP OUTLINE DIMENSIONS

SHEET INDEX:

SHEET																													
REV																													
SHEET																													
REV																													

TOLERANCES UNLESS NOTED OTHERWISE		DRAFTSMAN	
.X ±	90° FORMED ANGLES +2° −1°	CHECK	
.XX ±		MATL & FIN	
.XXX ±		APPL ENGR	
DEV		DEV ENGR	
[illegible]		PROJ ENGR	
[illegible] - SEE NOTE		PROD ENGR	
MATERIAL		RELIABILITY	

HONEYWELL INC. AEROSPACE AND DEFENSE GROUP

☐ AERO MPLS ☒ AERO ST PETERSBURG ☐ ORD SEATTLE ☐ ORD HOPKINS ☐ ORD MONTGOMERYVILLE ☐ S&R CTR MPLS ☐ RADIATION CTR BOSTON

OUTLINE DIMENSIONS, 10 CARD RNML HOUSING, HARD MOUNTED

SIZE	CODE IDENT NO.	DRAWING NO.
A	09128	NONE
SCALE	WT	SHEET

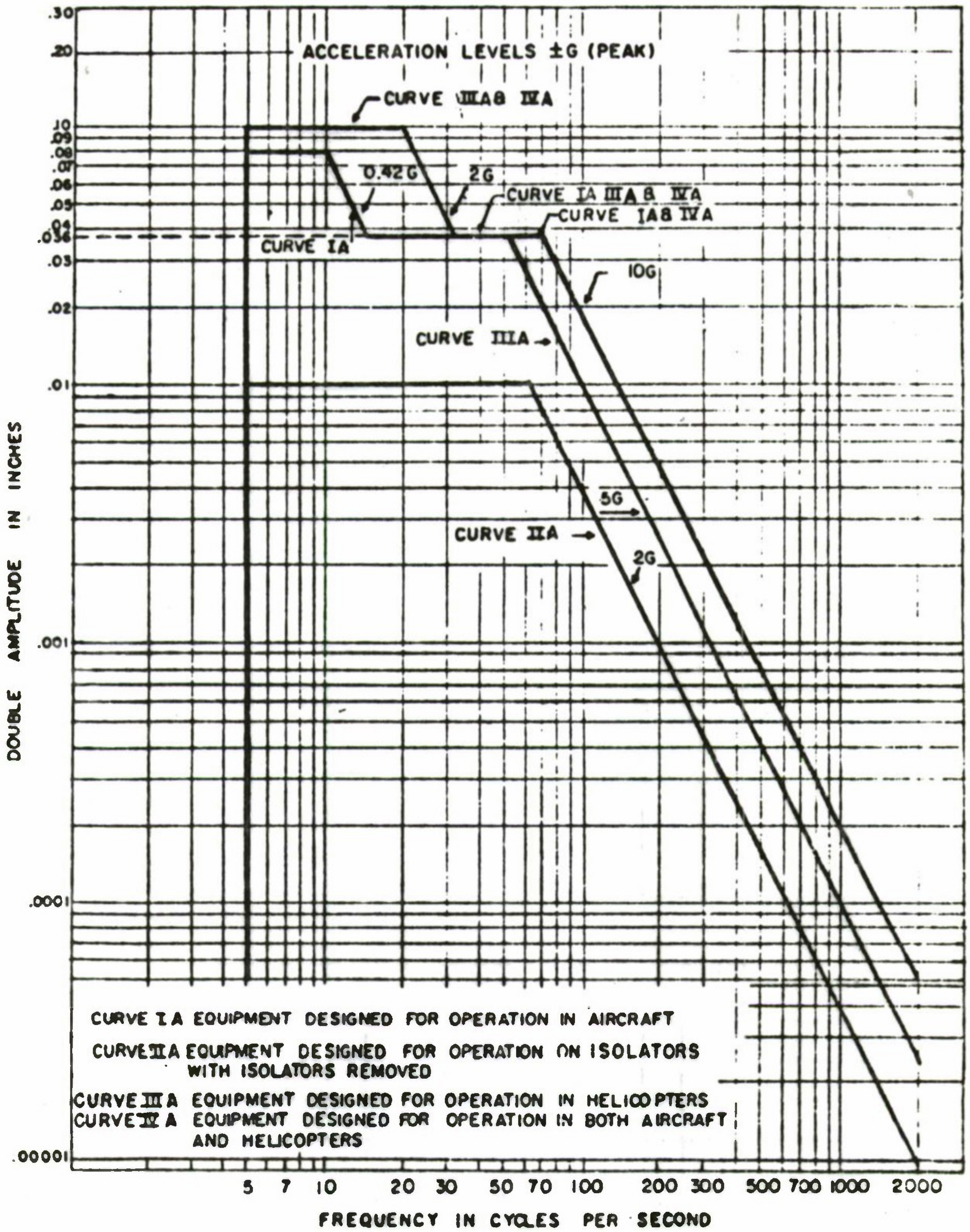


FIGURE 10. VIBRATION REQUIREMENTS

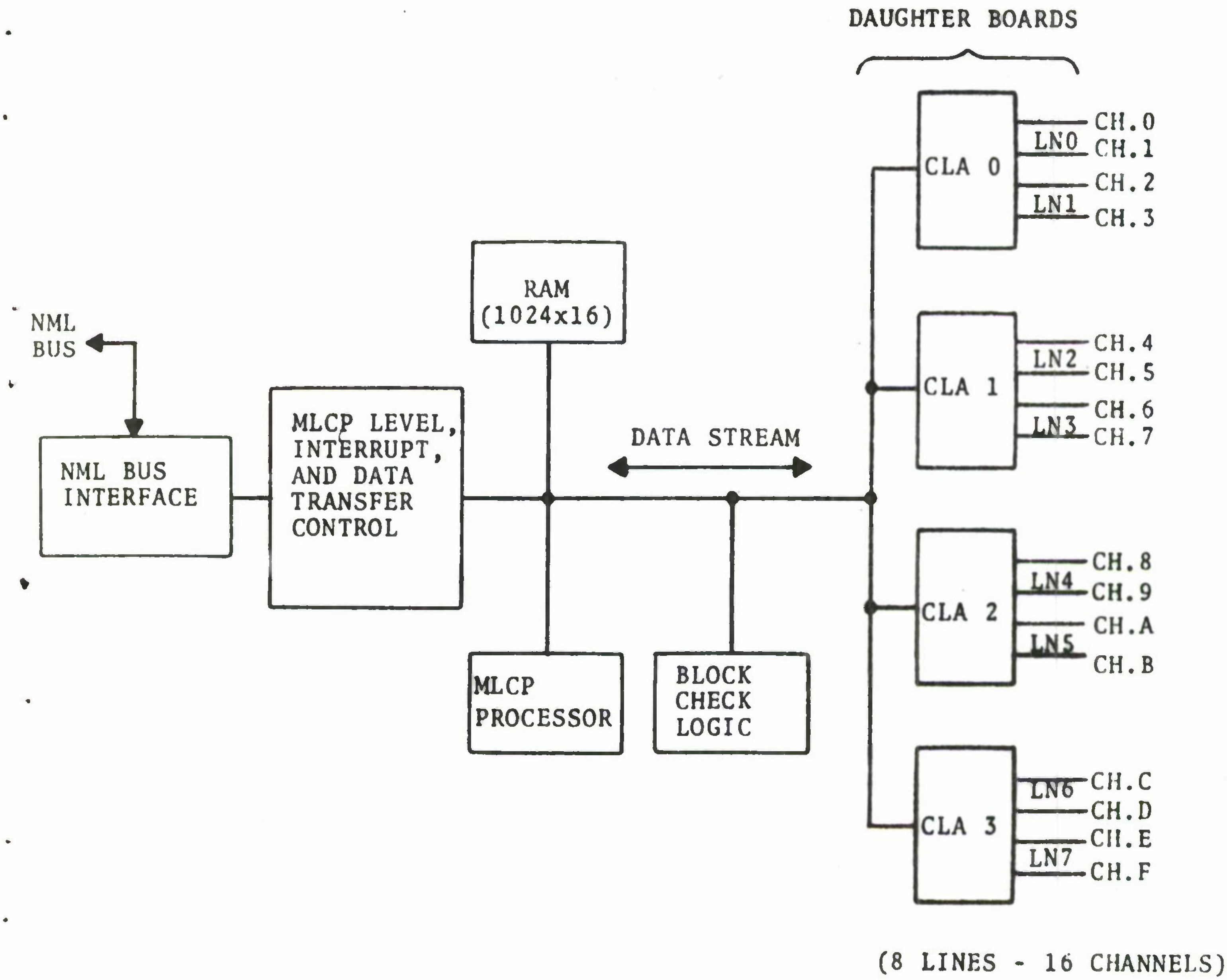
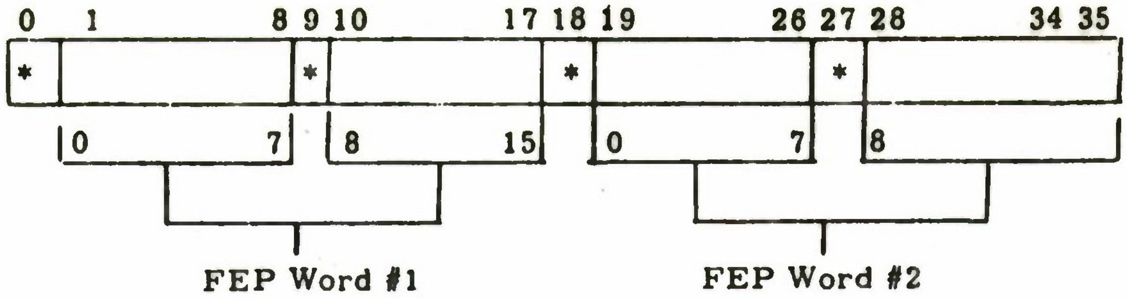


FIGURE 11. MLCP FUNCTIONAL BLOCK DIAGRAM



* Bits to be forced or interpreted as zero.

ASC II FORMAT

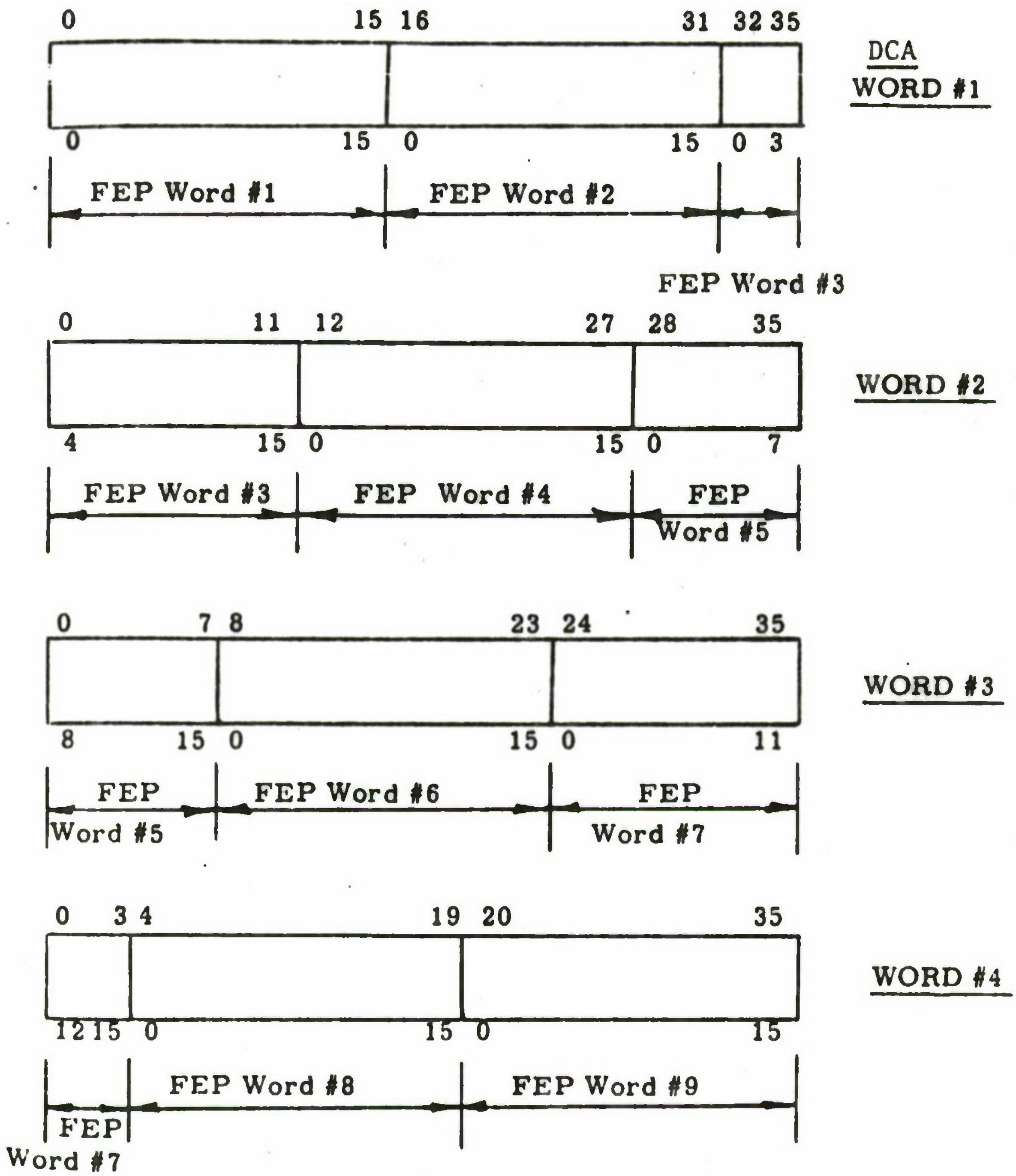


FIGURE 12. BINARY FORMAT

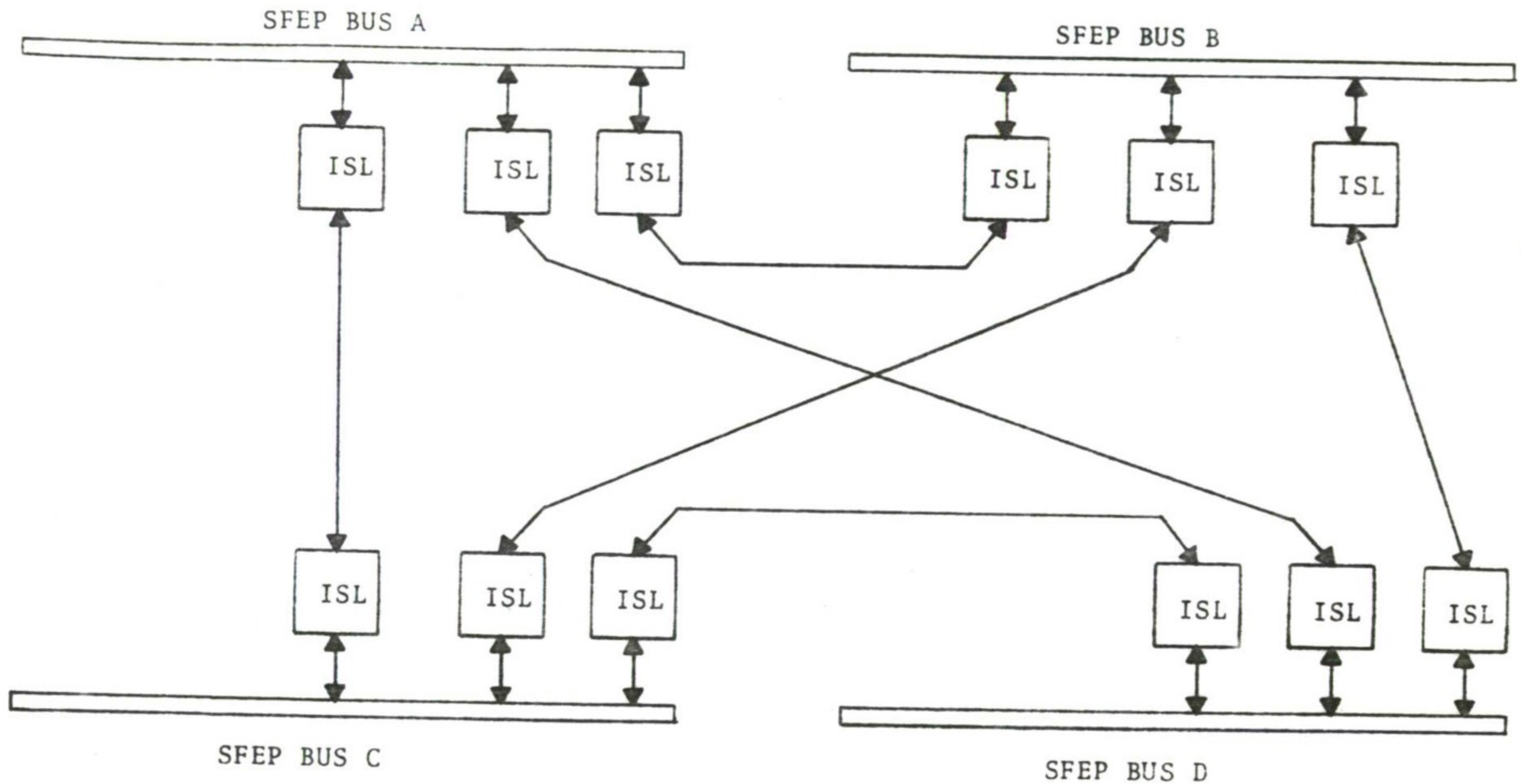


FIGURE 13. <u>ISL'S CONNECTING MULTIPLE BUSES</u>

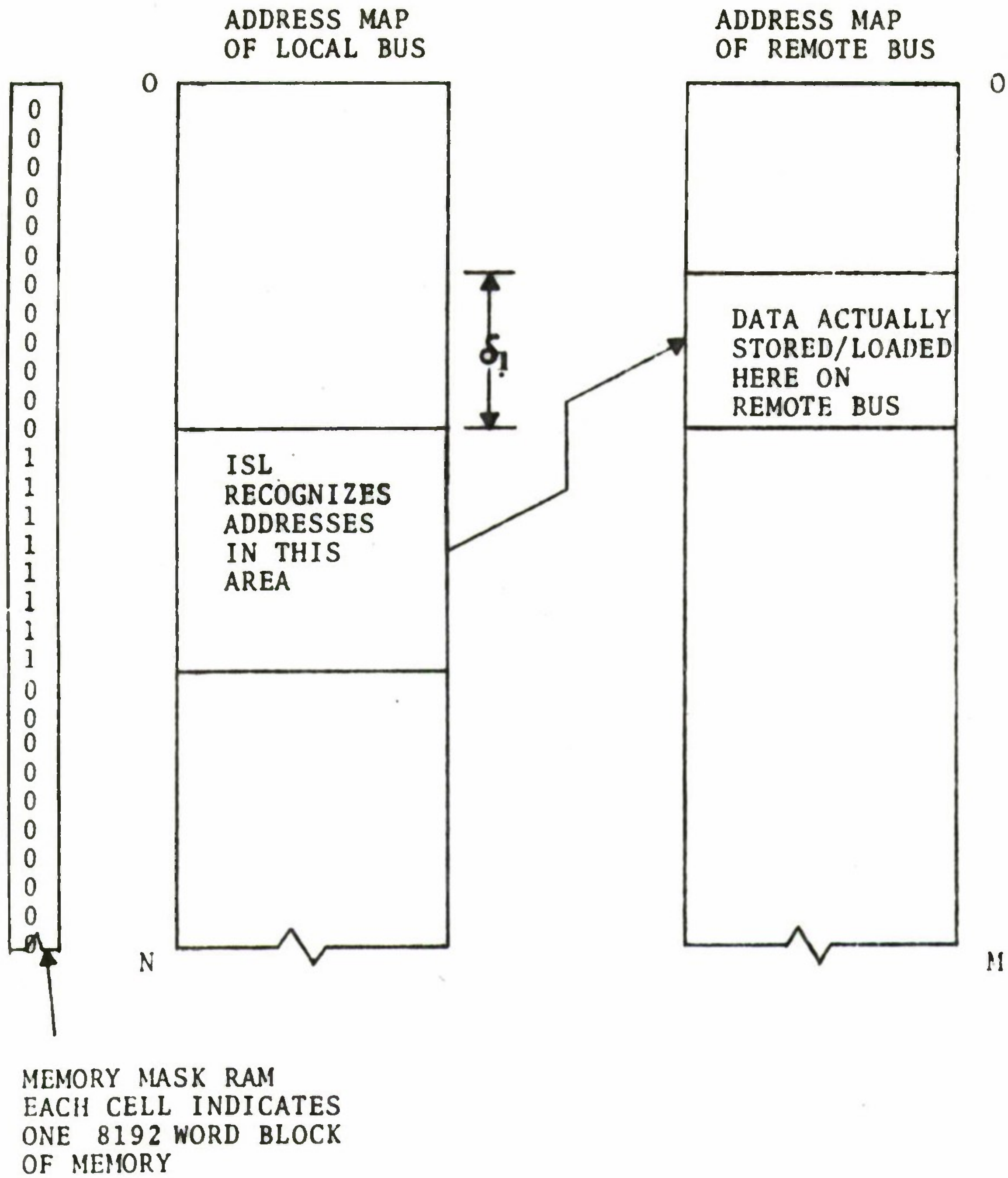


FIGURE 14. <u>ISL ADDRESS RECOGNITION AND TRANSLATION SIMPLIFIED</u>

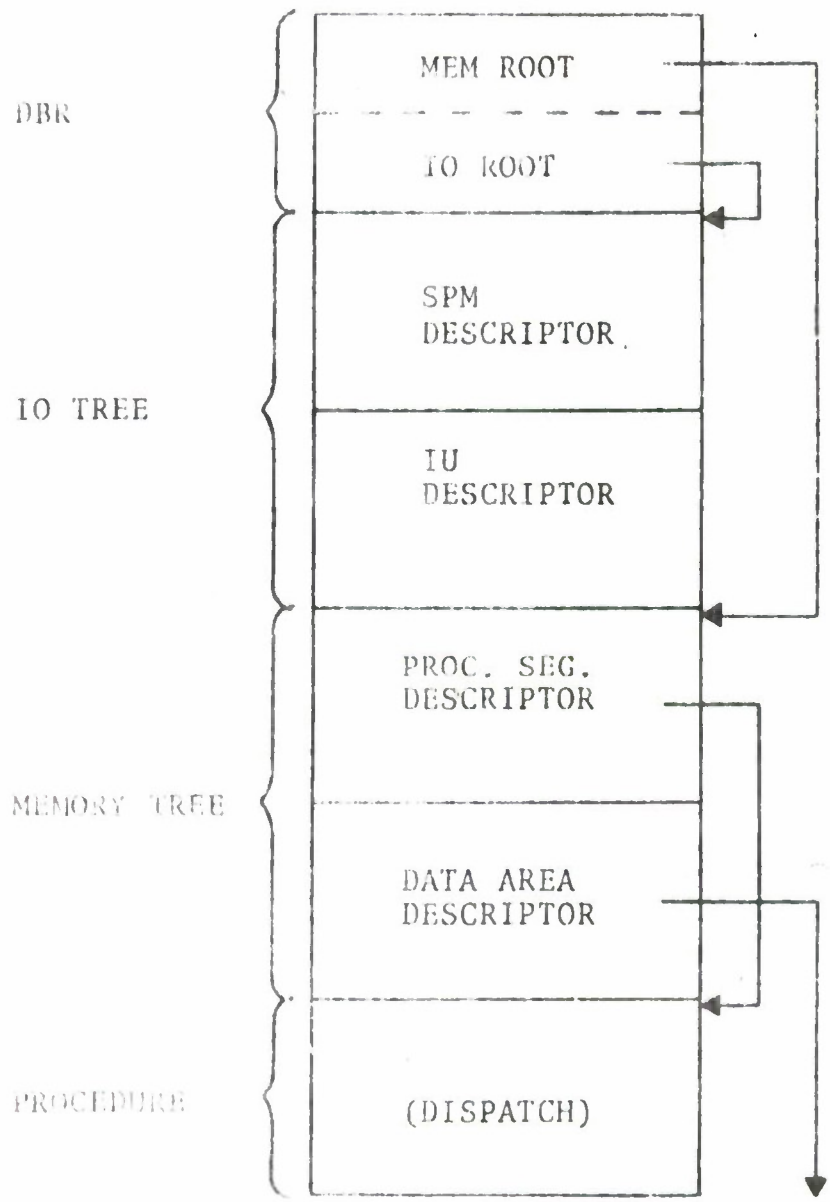
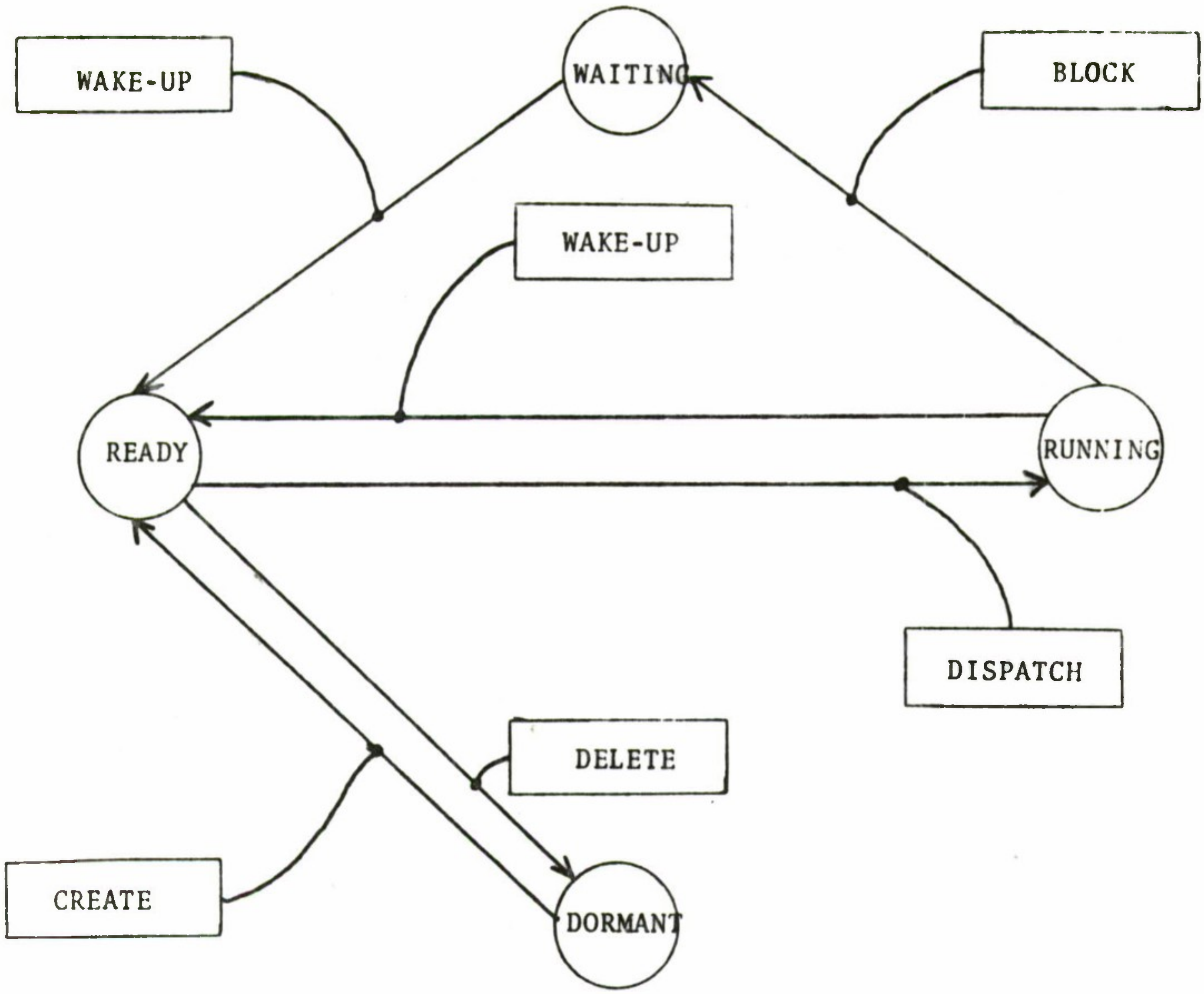


FIGURE 15. BOOTSTRAP



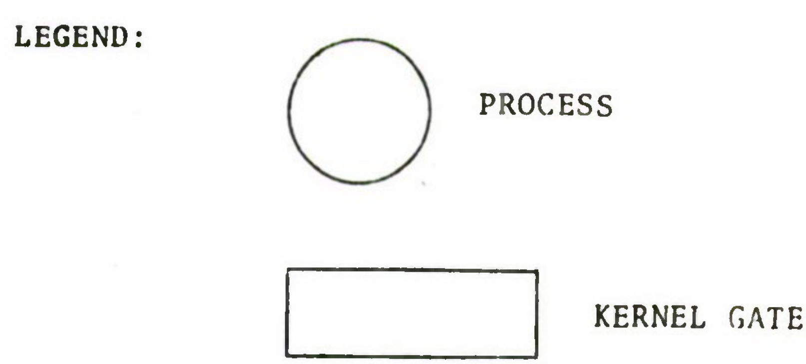


FIGURE 16. PROCESS STATES

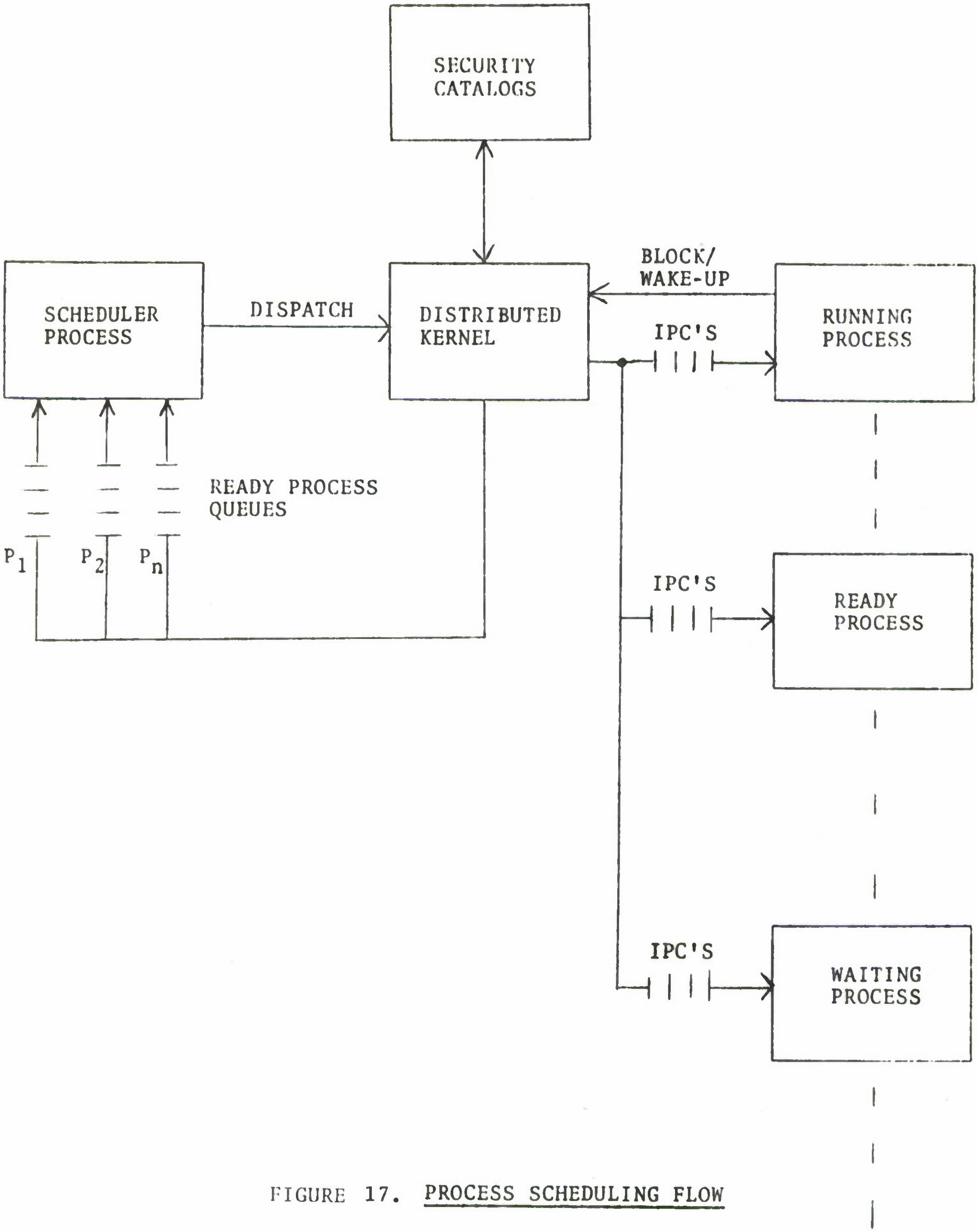


FIGURE 17. PROCESS SCHEDULING FLOW

APPENDIX

COMMENTS ON SFEP SUBSYSTEM SPECIFICATION

DATED 2 OCTOBER 1976

The "SFEP Subsystem Specification" presents what is basically a complete and sound description of the front-end processor subsystem of a secure computer system. The comments below note the deficiencies of the specification. In general, the specification lacks clarity and is inconsistent within itself and with the various specifications of the elements of the subsystem. Specific technical comments follow.

Para 3.1.1.1.1 -- The discussion does not incorporate integrity considerations. Subjects and objects should have integrity attributes. In this and the next section, the document should distinguish between formal kernel requirements as specified in the model, and design choices made to implement those requirements.

Para 3.1.1.1.2 -- Is it certain that the domain of the security kernel will be only ring Ø?

Para 3.1.1.2 -- Note that the kernel may have to provide specific functions for the support software. Figure 4 and the associated discussion do not make it clear that there is only one answering service, scheduler, and memory manager process, whereas there are many terminal handlers and perhaps more than one network interface manager.

The term "system high" near the bottom of page 13 seems out of place, especially since it has not been defined. Refer to the comments on the SFEP kernel (ESD-TR-76-359) regarding the use of untrusted processes to implement resource allocation policies for the kernel.

Para 3.1.1.2.1.1 -- The discussion of I/O on page 16 lacks an associated discussion of the verifiability issue to justify implementation of mapped I/O.

Para 3.1.1.2.2 -- The second paragraph seems to forget the fact that, since the kernel is responsible for defining the mapping between physical and virtual devices, it can easily prevent any process from choosing a virtual device address that is the same as the physical address of the SPM. No special function codes are necessary.

Para 3.1.1.2.2.1 -- Numerous undefined terms are used (e.g. R_{cur}, call bracket, call limiter, and direct page descriptor).

The reason for preventing outward calls is not to prevent a security compromise, but rather a software debugging convenience.

How the kernel can invoke untrusted resource allocation policy, when outward calls are prohibited, is not explained.

The problems with the proposed argument validation mechanism revealed at the SFEP preliminary design review renders the VALIDATE instruction useless. No comments hereafter will be made concerning discussions of that instruction in this document.

Para 3.1.1.2.2.5 -- The description of the SPM's action on a descriptor invalidation order appears questionable because it differs from typical FADS. Normally, after a descriptor is invalidated, it is not retrieved from memory until the descriptor is required for a memory access. If FADS entries must multiplex descriptor segment page table entries, it appears that the typical approach is more reasonable. The FADS resource could be used to maintain a descriptor for a different area of the virtual memory.

Para 3.1.2.1 -- The interface to the central host computer is not normally considered an interface to the "outside world". It is an interface external to the SFEP, but within the secure computer system.

Para 3.1.2.1.2 -- The first paragraph seems to imply that there can be only one IU per system, rather than one IU per IOM.

Para 3.1.2.3 -- It should be made clear that the trusted process-to-user process interface, though it may use the same IPC mechanism, is not the same as the process-to-process interface.

Para 3.3.1.2 -- The hierarchical design approach appears reasonable. However, it is not clear that implementation should be programmed starting at the highest level is the best approach since testing of the implementation cannot occur until all levels are tested. By beginning with the lowest level testing/debugging can occur at the completion of each level.

Para 3.3.1.2.3 -- Combining procedure and data segments that belong to the same ring and that have the same access controls is not acceptable since such a practice precludes reentrant code and requires multiple copies of the segments.

Para 3.3.7 -- This paragraph, entitled "Human Engineering", is missing, but is required by MIL-STD-490, 20.3.3.7.

Para 3.7.1.3.2.1 -- Are the 4 encodings of the ring fields really 3 encodings? Why is it necessary to use the type field to distinguish direct page descriptors from direct segment descriptors? Direct segment descriptors can only be at level 2, whereas direct page descriptors are always at level 3.

In the description of BASE, the phrase "modulo 128 or 13 bits" makes no sense unless one happens to know that 20-bit addresses are being used (which should have been stated). The factor of eight increase in resolution for indirect descriptors results in accuracy to 16 words or four descriptors. Apparently, this choice was based on the available field width and not on the most obvious and general case of single descriptor resolution. If a less-then-ideal choice has been made, the reasons should be stated.

The LIMIT field discussion leaves much to be desired. For direct page descriptors, is LIMIT the size of the page? (There is no use for LIMIT in this case). For direct segment descriptors is LIMIT the size of the segment? For indirect descriptors, is LIMIT the size of the segment or the number of descriptors? What is the meaning of LIMIT for level 1 descriptors?

Para 3.7.1.3.2.2 -- This discussion comes rather late in the document considering the numerous uses of the terms "page descriptor," "descriptor segment," etc., in prior sections that are defined here for the first time. The virtual address size is also given here for the first time though previous sections assume the reader knows what it is. The entire document suffers severely from this problem of terms used before being defined.

The discussion of Figure 7 would be easier to follow if the words in the text corresponded to those in the figure. For example, "LEVEL 1" should be used instead of "first level." At the bottom of page 68, the unpaged descriptor segment discussion should reference the appropriate portion of Figure 7 before the discussion, not after. In the figure, the notation "a + b" for offset into the unpaged descriptor segment implies addition of the two fields. The fields are concatenated--not added. Conceptually, the "SPM Flag" bit should be a control signal and not a bit on the address bus. The decision to use the address bus was made for convenience and hardware simplicity--it makes verification of hardware more difficult. Since the less than optional choice was made, that should be stated.

Para 3.7.1.3.2.3 -- The first paragraph seems to clarify the use of the LIMIT field that should have been discussed in 3.7.1.3.2.1. However, what is the use for a limit field in direct page descriptors? Note that all that is needed is a 9-bit limit field in the DBR describing the number of segments, and an 11-bit limit in the LEVEL 2 segment descriptor (direct or indirect) describing the number of words in the segment.

Para 3.7.1.3.2.4 -- The discussion of Reff is confusing. The main problem is with the statement "For each descriptor encountered between instruction fetch and operand fetch, a new Reff shall be computed..." Reff is to be calculated based on the R1 fields of all descriptors encountered during effective address calculation, but not during operand fetch. The value of Reff must not depend on the R1 field of the descriptor for the operand. This is because Reff < R1 must signal a fault for execute and call access. As written, Reff can never be less than R1 of the operand, thereby making it impossible to test for this fault condition.

Aside from the problems above, the notation is inconsistent: "REFF," and "Reff" are all used to mean the same thing. Also, in the description of permissions, no statement is made as to which R1, R2 and R3 fields are being used.

Para 3.7.1.3.3.3.1 -- In the second paragraph, it is stated that a new Reff (different from that used to fetch the I/O command?) must be computed when fetching the memory descriptor. As in 3.7.1.3.2.4, this is not true, since Reff is not a function of the location of the data.

There is no discussion of the action taken by the SPM when IOCT becomes greater than 16.

Para 3.7.1.3.3.3.2 -- On page 85, should it not be mentioned that the segment number to/from which data is to be transferred must also be saved by the SPM? If it is not, then the original segment number is forgotten when it is replaced by the device ID, as discussed on page 86.

The top of page 86 has a confused discussion of the U and M bits, because it doesn't make it clear whether the device descriptor or the memory descriptor is being referenced. Certainly, the U and M bits of the memory descriptors must be modified as well as those in the device descriptor (except the M bit is modified in the opposite sense).

The restriction to single-segment operations discussed on page 87 seems needless--the reader should be informed of the reasons.

Para 3.7.1.3.4.1 -- See comment on Para 3.1.1.2.2.5. An order to invalidate a descriptor usually does not cause the SPM to retrieve a new descriptor from memory to replace it. A new descriptor will be retrieved whenever the processor happens to next require a new descriptor. The discussions on page 89 and page 88 do not seem to agree.

Para 3.7.1.3.5 -- It was never made clear whether mapping is actually performed for normal transfer instructions. As a minimum, only the new virtual program counter needs to be loaded--the SPM need make no access to the destination until next instruction fetch. As an aid to debugging, however, it would be useful if the SPM at least validated the access to the destination before loading the PC, so that the location of an illegal transfer instruction will not be lost. The discussion of CALL should state that a CALL is exactly like a transfer with additional ring checks and changes of R_{cur}, rather than stating what CALL is not like. Are procedure entry points limited to the first location in the segment or the first location in any page of the segment?

On page 91, the "call bracket" is then stated as being R1 to R3, but the check for Reff > R1 is left out. The SPM specification included a check for Reff > R1. Whether or not the check is performed is not crucial. However, the selected alternative should be consistently presented.

The description of VALIDATE is rather terse and gives little information.

Para 3.7.1.3.6 -- The Reff < R1 check is omitted for execute violation. Throughout the document, the terms "trap" and "fault" are used interchangeably, due to the mixing of Level 6 and Multics terminology. One of these terms should be chosen and used exclusively.

What is the priority for trapping in the event that more than one trap condition occurs?

Para 3.7.1.4.2.2 -- Either the SPM or the MLCP must also be verified to insure that the direction of I/O--read or write--does not change.

Para 3.7.1.7 -- The system configurations possible with the ISL are not clearly enumerated. Is an SPM required on each bus in a multiprocessor system? In a multiprocessor system with an additional ISL used to extend the bus for more devices, is an SPM required on the remote bus? Doesn't the ISL for the remote bus respond to virtual addresses? Does the paragraph on page 103 apply only to multiprocessor systems?

Para 3.7.2.1.3 -- Besides performance, security requirements may also dictate that physical allocation policy be in the kernel, i.e., there may be no way to implement a given policy securely outside the kernel (e.g., quota in Multics).

Refer to the SFEP kernel comments for a discussion of issues in removing policy from the kernel (ESD-TR-76-359).

Para 3.7.2.2 and Para 3.7.2.3 -- Refer to comments on the SFEP kernel design (ESD-TR-76-359).

Para 3.7.2.4 -- The answering service does not fall into the category of applications software, since, as described here, it is part of the kernel. Naturally, demotion of the answering service to an untrusted process would be welcome.

Para 3.7.2.4.3 -- It should be noted that interfacing specifically with the ARPANET involves no trusted software, since the ARPANET is single level (unclassified). Secure networks require trusted software, at least to demultiplex the different security levels.

FIGURES -- The figures should be integrated with the document rather than placing them at the end (MIL-STD-490, 3.2.8.1).

MISSION

OF THE

DIRECTORATE OF COMPUTER SYSTEMS ENGINEERING

The Directorate of Computer Systems Engineering provides ESD with technical services on matters involving computer technology to help ESD system development and acquisition offices exploit computer technology through engineering application to enhance Air Force systems and to develop guidance to minimize R&D and investment costs in the application of computer technology.

The Directorate of Computer Systems Engineering also supports AFSC to insure the transfer of computer technology and information throughout the Command, including maintaining an overview of all matters pertaining to the development, acquisition, and use of computer resources in systems in all Divisions, Centers and Laboratories and providing AFSC with a corporate memory for all problems/solutions and developing recommendations for RDT&E programs and changes in management policies to insure such problems do not reoccur.